FPGA BASED IMAGE PROCESSING TOOLBOX USING CONVOLUTION BASED TECHNIQUES

Dissertation submitted to
Visvesvaraya National Institute of Technology ,Nagpur
In partial fulfillment of the requirements for the award of the degree

Master of Technology
in
VLSI Design

by

Vailla Spoorthi Priya
(MT23MVD007)

Under the guidance of

Dr.Sanjay B.Dhok

Center for VLSI and Nanotechnology
Visvesvaraya National Institute of Technology
Nagpur 440010(India)

June2025

FPGA BASED IMAGE PROCESSING TOOLBOX USING CONVOLUTION BASED TECHNIQUES

Dissertation submitted to
Visvesvaraya National Institute of Technology, Nagpur
In partial fulfillment of the requirements for the award of the degree

Master of Technology
in
VLSI Design

by

Vailla Spoorthi Priya
(MT23MVD007)

Under the guidance of

Dr. Sanjay B.Dhok

Center for VLSI and Nanotechnology
Visvesvaraya National Institute of Technology
Nagpur 440010(India)

June2025

©VisvesvarayaNationalInstituteofTechnology,Nagpur2025

Center for VLSI and Nanotechnology

Visvesvaraya National Institute of Technology, Nagpur

DECLARATION

I, **Vailla Spoorthi Priya**, here by declare that this dissertation titled "**FPGA Based Image Processing Toolbox using Convolution based Techniques**" is carried out by me in the **Center for VLSI and Nanotechnology** of Visvesvaraya National Institute of Technology, Nagpur. The work is original and has not been submitted earlier whole or in part for the award of any degree/diploma at this or any other Institution/University.

Vailla Spoorthi Priya
Date:

CERTIFICATE

This is to certify that the dissertation titled**"FPGA Based Image Processing Toolbox using Convolution based Techniques"**is submitted by **Vailla Spoorthi Priya (MT23MVD007)** fulfils the partial requirements for the **Master of Technology** degree **in VLSI Design** at VNIT Nagpur. The work is thorough ,complete and suitable for final evaluation.

Dr.S.B.Dhok
Head, Center for VLSI and Nanotechnology,
VNIT, Nagpur

Dr.S.B.Dhok
Head, CVN
VNIT, Nagpur

Center for VLSI and Nanotechnology

Visvesvaraya National Institute of Technology, Nagpur

NON-PLAGIARISMCERTIFICATE

Certified that M. Tech Dissertation titled **"FPGA Based Image Processing Toolbox using convolution based Techniques"** is submitted by **Vailla Spoorthi Priya(MT22MVD007)** has been checked for plagiarism using TURNITIN software and the overlap is found within prescribed limits. A summary of the report is given below,

ORIGINALITY REPORT

12%	8%	4%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

Figure : Plagiarism Certificate from TURNITIN Software

.

Vailla Sporthi Priya

Date:

Dr. Sanjay B . Dhok
Supervisor

ACKNOWLEDGMENT

Despite my name being the only one on the cover, a lot of others helped develop this research. My sincere gratitude goes out to everyone who made my study possible.

My deepest gratitude is to my Guide, **Dr. Sanjay B. Dhok.** And Co Guide **Dr. G.CPatil .**I consider myself extremely fortunate to have a mentor like him, who gave me the freedom to conduct independent research while simultaneously giving me the guidance I needed to get back on track when I made mistakes. I had some enlightening conversations and constructive criticism during my project that got me to reflect and helped me focus. His understanding and support allowed me to get through a number of challenging circumstances and finish my project.

Additionally, I would like to thank **Dr. R.M.Patrikar, Dr. R. B. Deshmukh, Dr. P.Vasu,** and **Dr. Nikhil Deep Gupta** for their constant advice that improved my Project work .I appreciate my classmates support in keeping me motivated while I worked on my project and my sincere regards for The Institute Director, **Prof. P. L. Patel**; Dean (Acad**.) Prof. V. R. Kalamka** and Associate Dean (Acad - PG/PhD) **Prof. R. V. Taiwade** for their unwavering support for student academic requirements.

Above all ,none of these achievements would have been attainable without the love and patience of my family. I want to extend my sincere appreciation to my family.

Vailla Spoorthi Priya

ABSTRACT

Recent improvements in FPGA technology have opened up new possibilities for implementing computationally expensive image processing techniques in embedded devices. These reconfigurable platforms have inherent parallelism benefits over standard CPUs, although existing solutions frequently focus on specific application domains. There is still a strong demand for flexible frameworks that can enable numerous processing methodologies while effectively managing constrained on-chip resources.

This paper describes a comprehensive framework of real-time processing images on the Basys3 Xilinx platform[18],which combines Verilog hardware design alongside Python preprocessing tools. The system systematically stores binary-converted data in Block RAM with efficient access patterns allowing convolution operations, allowing for both fundamental changes (brightness, gray scale conversion) and complex approaches (blurring, detection of edges, sharpening). Processed outputs are displayed in real time on the VGA display and Python scripts generate the coefficient files required to appropriately initialize FPGA ram with picture data.

The framework highlights FPGAs' great potential in applications that require high-speed, minimal latency image processing. Implemented convolution methods are important in visual analysis because they assist with reducing noise and highlighting features. Since object recognition, autonomy and medical use need it, extraction must be performed. imaging [1]. Software designed as modules can be built on with future additions. modifying algorithms or using specific filtering tools suited to specific industries. It helps ease the differences between hardware description languages. The use of advanced programming tools means the solution is both straightforward and very powerful. For running detailed analysis that requires high computation power on small devices platforms.

LIST OF FIGURES

Fig1.1Latency vs Development time.....3

Fig2.1Basys 3 Components Description 11

Fig2.2 Image Data seen as R, G, B Channels 13

Fig2.3Contents of a .COE File..... 16

Fig2.4Function defined in OpenCV Python 16

Fig2.5Single Port RAM Memory type 18

Fig2.6 Image showing the Hsync 19

Fig 3.1.Block Diagram of the FPGA Implementation 24

Fig3.2.Generated .Coe File of the input image 28

Fig3.3Configure IP Catalog in Vivado..... 29

Fig3.4 Instantiate Memory in RAM..... 29

Fig3.5Load the generated .Coe file 30

Fig3.6Image module Instantiation..... 30

Fig3.7RTL Schematic of inputs..... 31

Fig3.8RTL Schematic of the outputs showing R,G,B Channels 32

Fig3.9 Complete RTL Schematic of the design 33

Fig4.1Synthesized design of the RTL 36

Fig4.2Sucessfull generation of the bit file.....36

Fig4.3Basys-3 FPGA Board Implementation and Bit file Generation.....38

Fig4.4RGB to Gray Scale Conversion 38

Fig4.5Increase in the Brightness of Image 39

Fig4.6 Colour inversion Filter..... 39

Fig4.7Average Blurring Filter.....40

Fig4.8 Emboss Filter 40

Fig4.9Sharpen Filter.....41

Fig4.10Sobel Edge Detection 41

Fig4.11Motion Blur Filter 42

Fig4.12 Edge Detection Filter.....42

Fig4.13 Power contribution and report on Design 43

Fig 5.1 Edge Detection in mechanical parts inspection 45

Fig 5.2.Lane Detection with the Edge Detection 46

LIST OF TABLES

Table4.1 Input Parameters for Image 35

Table 4.2 Input selects and Corresponding Filters.....37

ABBREVIATIONS

VGA	Video Graphics Array
FPGA	Field Programmable Gate Array
VLSI	Very-large-scale Integrated
BRAM	Block RAM
RAM	Random Access Memory
BMG	Block Memory Generator

Table of Contents:

ABSTRACT i

LIST OF FIGURES.....ii

LIST OF TABELS……..…….iv

ABBREVIATIONS……...v

Chapter1: INTRODUCTION

1.1 Background……2

1.2 Motivation…….4

1.3 Objective……...5

1.4 Report Organization……..6

Chapter2: LITERATURERE VIEW

2.1 FPGA Basic Concepts……..9

2.1.1 Core Principles of FPGA Technology…….9

2.1.2 Basys 3 Implementation…….10

2.2 Image Data Processing Fundamentals……....12

2.2.1 Structure of an RGB Image……12

2.2.2 OpenCV (Open Source Computer Vision Library) ……...14

2.2.3 COE File Generation using OpenCV Python…….15

2.3 Design Architectures……..17

2.3.1 Block Memory Generator……..17

2.4 VGA Interface and Image Buffer……. 18

2.5 Image Data Processing and Edge Detection……..……...19

2.5.1 Sobel Edge Detection……..……...20

2.6 State of Art…… 21

Chapter3: CIRCUIT LEVEL DESIGN

3.1 Top Level Circuit……. 23

3.1.1 Explanation of the Block Diagram…….. 25

3.2 Image File Generation(.Coe file)……... 26

3.2.1 OpenCV Python Scripting……..... 26

3.3 Creating RAM/ROM in Vivado (Memory instantiation)……..… 28

3.4 Image Processing Unit……... 31

Chapter4: IMPLEMENTATION AND RESULTS

4.1 Circuit Level Design Results……..... 35

4.1.1 Synthesis Design Implementation…… 36

4.1.2 Bit file Generation…… 36

4.2 Output simulations Based upon the Select line…… 37

4.3 Simulation Results on the Basys 3 Board…… 37

4.4 Results……38

4.5 Power Analysis and Summary……. 43

Chapter5: CONCLUSION AND FUTURESCOPE

5.1 Conclusion…….. 45

5.2 Future Scope……45

REFERENCE……...47

1

INTRODUCTION

1.1 Background

The Field-Programmable Gate Array (FPGA) is a major development in logic circuits in reconfigurable computing technology, engineers are given freedoms they have never had digital circuit implementation can be done in many flexible ways. These circuits are built to include logic that is arranged in ways that are configurable depends on the needs routing resources makes it possible for hardware designers to build their own special. In FPGAs, processors run instructions in parallel which supports spatial computing. Programs that let groups of operations to take place in sync over the device. This fundamental specialized architecture greatly improves how computation is carried out Both intense workloads in parallel and easy flexibility are provided by them. So that it can meet newer algorithmic needs by changing its structure [9].

Processing digital images uses computers in ways that are ideally suited for FPGAs. General-purpose software struggles with high-resolution images because they must take each step in order which slows them down. On the other hand, FPGAs make it possible to set up pipelines that have several parts acting on different sections of the image at the same time. Because of this architectural choice, applications that require handling large volumes of data in a fast way work much better such as medical diagnostics[11], industrial inspection[2], and autonomous navigation systems[5]. The way FPGAs are built guarantees that processing times will not vary, even with heavy workloads which is important for reliability in real-time situations.

How FPGAs are designed today has changed a lot shifting from purely describing hardware in a single language to including high-level synthesis techniques. Now, FPGAs are easier for anyone to use as software programming languages are used for writing algorithms and the FPGA hardware design is handled automatically. Researchers and engineers are now able to use advanced image processing techniques. This contemporary evolution of FPGA from the exclusively HDL approaches has been making the configuration of the board quite and easy.

Algorithms—including spatial filtering, geometric transformations, and morphological operations—without extensive hardware design expertise. This evolution parallels similar trends in software engineering where abstraction layers have consistently enhanced productivity and accessibility[7],though with the added dimension translating algorithmic intent into physical circuit configurations.

The convergence of FPGA technology with artificial intelligence methodologies has catalyzed innovations in computer vision applications. The use of FPGA with artificial intelligence. has pushed for progress in computer vision applications. Specialized processing features in today's FPGAs support convolution neural networks and other deep learning systems which can be implemented directly within the hardware. Systems that use programmable logic and embedded processors allow vision systems to combine the best features of software and hardware techniques. They make it possible to organize processing steps like pre-processing photos, drawing out features and using a neural network. by integrating the system, we try to use less energy, send less data and work more efficiently[3], minimizing data transfer overhead and power consumption while maximizing computational efficiency.

Even though image processing on FPGAs comes with distinct benefits, there are still problems that should be taken into account. Because of the special design features needed for hardware, efficient algorithms and integration, this area requires skills from several disciplines that are normally not needed with software-only products. When working on highresolution pictures, the limited memory bandwidth can cause significant performance problems, so advanced strategies for managing data are required. In addition, there are major differences between verification and validation approaches and software engineering, as simulation is done at several levels of abstraction. While these, the performance density, better energy efficiency and precise processing abilities FPGAs are making them more important in advanced imaging since the tools used for their development are getting easier to use and more efficient.

1.2 Motivation

There is a need for strong and reliable systems to process images has become more important as new developments happen in digital processing quickly. Complex image processing algorithms[6] now be executed at previously unheard -of speeds thanks to developments. These algorithms can now run as large and complex as before remarkable speeds now possible because of advancements in field-programmable gate array (FPGA)technology over the years. The pattern points out that strong imaging frameworks help manage live images by using the parallel processing features of FPGAs.

Because visual information is growing in apps like autonomous Because vehicles are used in medical imaging, it is important to have image processing solutions that handle regular tasks as well as techniques such as edge detection and convolution filters. It is obvious from Figure 1.1 that these tasks are becoming more and more complex, so a system must be capable of working at high speeds, offering detailed resolution and low latency.

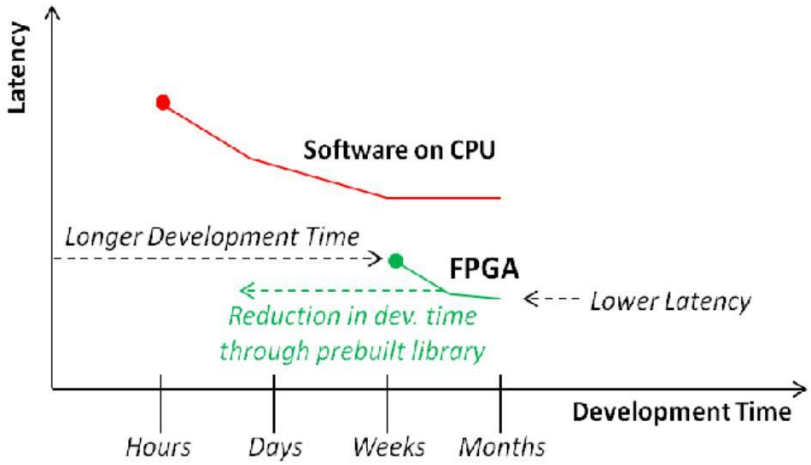

Fig:1.1.Latency vs Development time

The graph highlights that FPGAs can carry out a lot of processing very fast. due to their flexibility, they are good for situations that call for prompt responses. The ability to change with new conditions. Such abilities are vital when decisions have to be made very quickly such as in live monitoring and fast scene analysis.

The Image Processing Toolbox, built on FPGAs, is especially useful where power use is important such as in portable and battery-powered devices. The project wants to achieve a solution that can handle modern image processing tasks both efficiently and smoothly by using efficient algorithms and optimized memory management.

The use of convolution neural networks (CNNs) in tasks is another advancement. New capabilities for advanced image analysis are made possible by the FPGA framework. With CNNs able to discover relevant features and recognize patterns, they allow more detailed analysis of images. This project works to join CNNs with traditional image processing, forming a set of tools useful for multiple users and uses. This project seeks to harness the power of CNNs alongside other sides [3].

The motivation behind developing Why the FPGA-based Image Processing Toolbox was created has come about because of the rise in demands for speed, efficiency and versatility in image processing tasks. By relying on the strengths of FPGA technology and integrating advanced algorithms, this project aims for contribution in significant to the field of digital image processing, paving for the way to innovative applications across diverse industries.

1.3 Objective

The poster outlines a project to put into practice various image FPGA (Field Programmable Gate Array) technology is used in our processing operations. The purpose is to use embossing, blur, sharpen and edge detection on images to enhance their appearance. This method ensures that changes and feedback can be offered without any delay which is necessary in situations that must react quickly. The versatility of the FPGA technology has advantages over the software technologies.

To facilitate the Processing images is made easier in the project by converting them into a format (.coe) that can be read and used by an FPGA. This process is needed so that the FPGA can efficiently do the desired image processing. Along with it is important for users to be involved, so project is working on an interface where they can select from a

range of specific image processing functions. This feature enhances flexibility of these systems, allowing users to edit and customize the processing according to their needs.

The last part involves connecting the images to a VGA interface and displaying what is seen display outcomes of image processing clearly to everyone involved. This proves that FPGA technology is suitable for real-time processing of images and also demonstrates the results of the solutions set up. Overall, the project aims to bridge the gap between theoretical image processing techniques and practical implementation in hardware.

1.4 Report Organization

The report is organized into 5 chapters as discussed below.

Chapter1

Discusses the introduction of the project, motivation for doing this research work, project objectives, and thesis organization.

Chapter2

Provides a basic understanding of the Image data Processing and different types of architectures for implementing different filters and edge detection techniques. The implementation of the FPGA based image processing toolbox with the Xilinx Vivado is covered. The different edge detection technologies implementation is also discussed.

Chapter3

Presents the top-level block diagram of the tool box and the circuit-level designs of individual components involved in the design. The schematic level simulations and RTL coding of these components with the help of Xilinx Vivado tool are also discussed.

Chapter4

This chapter shows the final implementation and the results of the toolbox and various applications of the filters for our required specifications.

Chapter5

The final chapter concludes with a summary of the work presented and the results. Also discussed the future outlook of our research study.

2

LITERATUREREVIEW

In this section, the basic fundamentals of the FPGA and image processing tools are discussed. A thorough explanation of the edge Detections and various filters is provided.

2.1 FPGA Basic Concepts

2.1.1 Core Principles of FPGA Technology

Field-Programmable Gate Arrays belong to a unique type of semiconductor. hardware that can be set up in different ways after it is built. Conventional processors do not offer this flexibility, since these chips have programmable components that engineers can use to create different digital features. Because of this distinction, a hardware-based style of computing offers many benefits in some specialized cases. This Fundamental distinction enables a Hardware oriented approach with several inherent advantages in the other designs.

FPGA architecture includes several configurable parts arranged in a grid. Logic cells that do Boolean functions, paths that move signals between components, memory areas and separate processing units are part of this. During configuration, the developer writes a circuit description in VHDL or Verilog which tools then change into data telling the device how to operate its components defines the behavior of each programmable element within the device. These include logic cells capable of implementing Boolean functions, interconnection pathways that establish signal routing between components.

The strengths of FPGA technology become FPGAs work best where loads are distributed across many processing elements, exact timing is needed or there are required unique interfaces. Parallel implementation of algorithms can be made much faster by their ability to process different streams of data simultaneously. Therefore, using them is especially popular in signal analysis, video processing, cryptography and creating special protocols, as common processors may not perform as well in these areas. [12].

2.1.2 Basys 3 Implementation

Basys 3 is designed as a simple platform for working with FPGAs from the Artix-7 series produced by Xilinx. It is built mainly for classrooms and prototypes, as it is simple to use and also has the features needed for effective project development. It has been built as a single design that merges the main FPGA with many ways to interact and connect. This is Designed primarily for educational environments and prototype development.

The platform features a comprehensive peripheral set including:

i. User input mechanisms comprising sixteen toggle switches and five momentary push buttons.
ii. Visual output elements consisting of sixteen individual LEDs and a four-digit multiplexed seven-segment display.
iii. Communication interfaces including USB-UART conversion circuitry.
iv. Display connectivity via a standard VGA port.
v. Expansion capabilities through peripheral module connectors.
vi. Integrated USB programming interface for streamlined development.

Basys 3 projects go through a number of different phases in their development cycle. First create the circuit using Vivado, then check its function through simulation and only then proceed to synthesis by turning the design into a form that can be implemented on the target FPGA[18]. After implementation, timing analysis are done, the tools produce a configuration bit stream used to program the device via its USB interface [18].This implementation set to a functional verification through simulation tools before proceeding to synthesis- process of translating abstract descriptions following implementation and timing analysis.

Educational applications frequently begin with straightforward implementations such as combinational logic circuits, sequential systems, or basic controllers that utilize the onboard interaction elements. As proficiency increases, more sophisticated projects become

feasible, including custom processors, digital signal processing implementations, or interfacing with external hardware through the expansion connectors.

Though its main purpose is educational, the Basys 3 gives you a good deal of computational power. Resources consist of about 33,000 configurable logic cells and some dedicated processing elements. It supports building digital systems that are not overly complex, giving students and developers opportunities to test new ideas in digital design with popular development technologies.

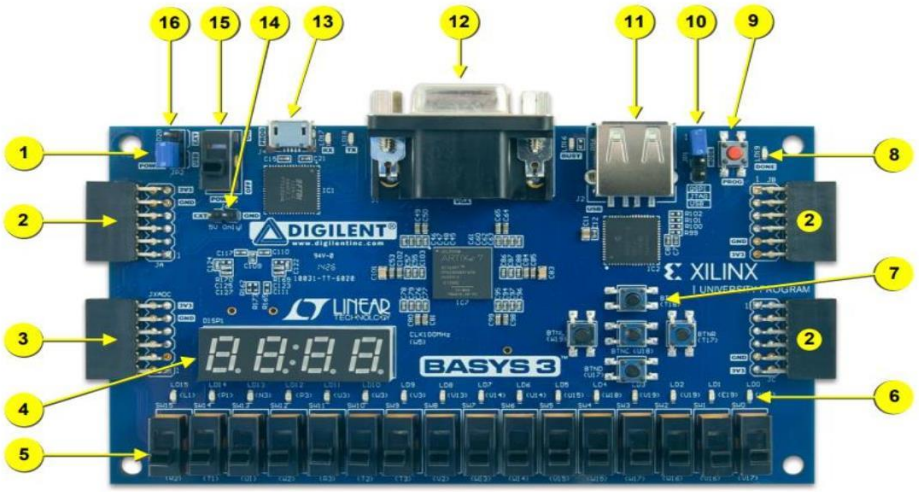

Figure 1. Basys 3 FPGA board with callouts.

Callout	Component Description	Callout	Component Description
1	Power good LED	9	FPGA configuration reset button
2	Pmod port(s)	10	Programming mode jumper
3	Analog signal Pmod port (XADC)	11	USB host connector
4	Four digit 7-segment display	12	VGA connector
5	Slide switches (16)	13	Shared UART/ JTAG USB port
6	LEDs (16)	14	External power connector
7	Pushbuttons (5)	15	Power Switch
8	FPGA programming done LED	16	Power Select Jumper

Fig2.1Basys 3 Components Description

Specifications:

- 33,280 logic cells in 5200 slices (each slice contains four)
- 6-input LUTs and 8 flip-flops)

- 1,800 Kbits of fast block RAM
- Five clock management tiles, each with a phase-locked loop (PLL)
- 90 DSP slices
- Internal clock speeds exceeding 450MHz
- On chip analog-to-digital converter (XADC)

The Basys 3 additionally provides an upgraded set of ports as well as peripherals, which are indicated in the Fig 2.1:

- 16 user switches.
- 16 user LEDs.
- user push buttons.
- 4-digit, 7-segment display.
- Three Pmod ports.
- Pmod for XADC signals.
- 12 bit VGA output

Features include a USB-UART bridge, serial flash, and a powerful USB-JTAG port for programming and communication with FPGA, USB HID host to Mouse, Keyboard, and Memory Stick.

2.2 Image Data Processing Fundamentals

2.2.1 Structure of an RGB Image

Digital images are essential to modern technology, acting as the principal form of representation of information in a variety of devices and applications. Digital images are essentially created up of pixels—small, separate components that work together to generate a

complete picture. Each pixel includes color information, and because of its great density & human vision perceives a continuous picture when seen from a standard distance.

The RGB color system is the most widely used approach for describing color in digital photographs. RGB refers to red, green, and blue, the three basic hues of light. A wide range of colors can be achieved by changing the intensity each of these hues. Each pixel in a picture that uses RGB is represented as three values: red, green, and blue as seen in fig 2.2 These numbers typically vary between 0 and 255, enabling 256 level of intensity every channel. When combined, this yields nearly 16 million color choices, creating a rich and comprehensive color palette for pictures[1].

Digital cameras and scans use color filters on their sensors to acquire color information. The Bayer filter pattern is a popular configuration that employs a grid of blue, red, and green filters across the sensor's photo sites. This design usually has twice more green filters as the blue and red ones, reflecting a human eye's increased sensitivity for green light. After recording the image, the image sensor creates a mosaic with color data, that is subsequently processed using a technique known as demosicing. Demosicing interpolates missing color information to each pixel, resulting in an image with full color from the incomplete data provided from the sensor's filter array.

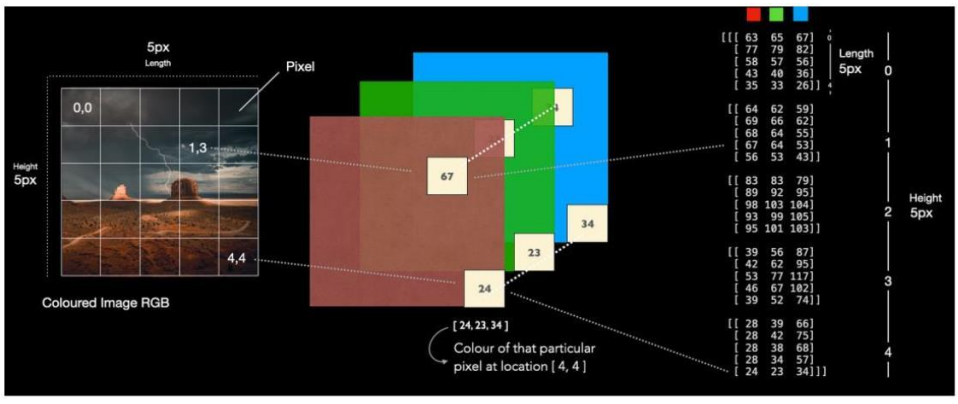

Fig 2.2: Image Data seen as R, G, B Channels

After being taken, digital images are kept and can be sent in various computer file formats. For example, JPEG, PNG and BMP are included. They explain the way data from pixels is arranged and reduced in size. As an illustration, in a 24-bit picture, each pixel's colour is broken into 8 bits of red, 8 bits of green and 8 bits of blue all together. The encoded data is put into a file system that maintains it safely and transmits it properly. As soon as an image appears on a screen, the encoded data is read and the pixel values are used to make the pixels on the screen light up, showing the image just as it was originally shot. The data is subsequently encoded in a file system suitable for preservation or transmission.

2.2.2 OpenCV (Open Source Computer Vision Library)

OpenCV is a powerful open-source library built for actual time computer vision and image processing. With Python bindings, it provides a comprehensive set of tools for working with photos and movies, including object identification, image manipulation, and feature extraction.

It is an extensive open-source library of computer vision, artificial intelligence, and image processing. It now plays a significant part in real-time operation, which is critical in modern systems. It allows you to process photos and movies to recognize items, faces, and even human handwriting.

Python, when combined with other libraries such as NumPy, can parse the opencv array format for analysis purposes. To identify patterns and its many aspects, we use vector spaces and apply mathematical operations to them.

The initial OpenCV version is 1.0. OpenCV is published under a BSD license, therefore it is free for academic as well as commercial use. It is compatible with Linux, Windows, Mac OS, iOS, and Android and provides interfaces in C++, Python, C, and Java. When opencv was created, the primary focus was on applications that operate in real time for computational efficiency. Everything is written in efficient C/C++ to make use of multi-core processing.

Bellow is the command of the for install OpenCV

```
pip install opencv-python
```

OpenCV enables you to do numerous operations on images.

- **Read the image**. OpenCV allows you to read an image from a file or directly from the camera and make it available for further processing.
- **Image enhancement**: You can improve an image by altering its brightness, sharpness, or contraction. This helps you visualize the image's quality.
- **Object detection**: As shown in the image below, OpenCV can detect objects such as bracelets, watches, patterns, and faces. This can include recognizing people, shapes, or even objects.
- **Image Filtering**: One can adjust the image by using filters like blurring or sharpening.
- **Saving the Changed photos**: After processing, you can save the modified photos for further study.

2.2.3 COE File Generation using OpenCV Python

A.coe file is a text document which specifies the memory module's initial contents. In image processing, the.coe file is used to fill the BRAM with pixel information from an image. The file starts with a header which specifies the radix (number base) to the data, then the memory initialization vector, that lists the pixel values. Depending on the image format, each value represents either the intensity as well as color component of a pixel. These values are placed inside the BRAM during the implementation of the FPGA configuration, giving the data required for image processing[13].

The COE file contains a list of the values for every address in memory. Here is the contents of a COE file as show in the Fig 2.3.

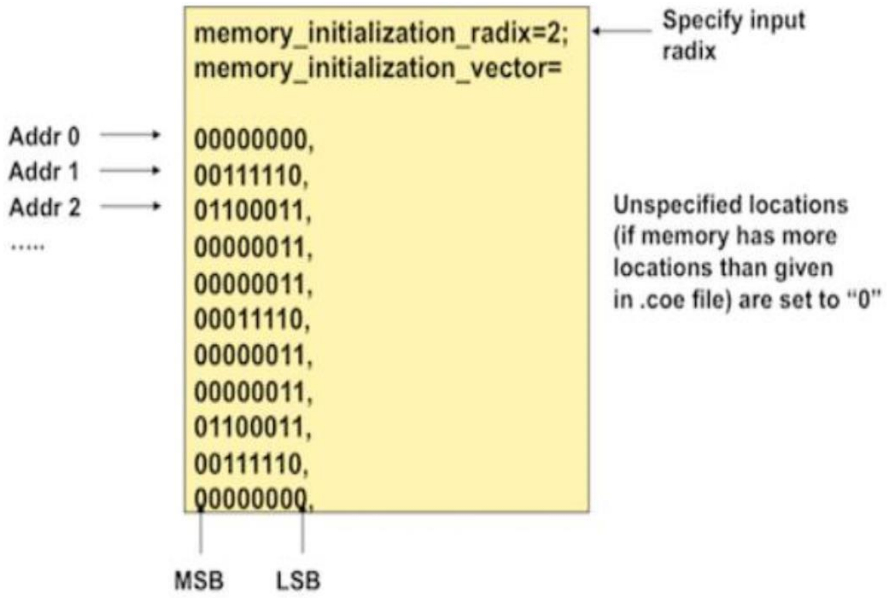

Fig.2.3. Contents of a .COE File

Bellow is the function to read is the image from jpg to .coe format as seen in fig.2.4

```
def write_coe_file(filename, data):
    with open(filename, 'w') as f:
        f.write("memory_initialization_radix=10;\n")
        f.write("memory_initialization_vector=\n")
        for row in data:
            for pixel in row:
                f.write(f"{pixel},\n")

# Write the COE file
write_coe_file('image.coe', img_normalized)
```

Fig.2.4 Function defined in OpenCV Python

2.3 Design Architectures

2.3.1 Block Memory Generator(Creating RAM/ROM with Vivado)

The BMG v8.4 represents an IP-based core developed by Xilinx that allows for the development of blocks RAM (BRAM) modules inside an FPGA. The BRAM modules are required for storage data such as pictures, which may then be retrieved during processing activities. The BMG supports the setup of a variety of characteristics, including memory depth, and width, and initialization choices. For image processing operations, a single-port ROM design is commonly employed, in which the FPGA reads picture data sequentially[18].

This Block Memory Generator fundamental creates custom memories from block RAM primitives for Xilinx® FPGAs. The core optimizes memory by organizing block RAM primitives according to user preferences and automating their instantiation and concatenated. The Vivado Design Suite IP Catalogue allows for core configuration and rapid generation of optimized bespoke memory solutions.

Memory Type

- Single-port RAM
- Simple Dual-port RAM
- True Dual-port RAM
- Single-port ROM
- Dual-port ROM.

Optimizations are implemented inside the core to reduce overall resource usage for each setup .Figure 2.5 illustrates how single-port RAM provides both write and read access to memory via a single port which is used in the project.

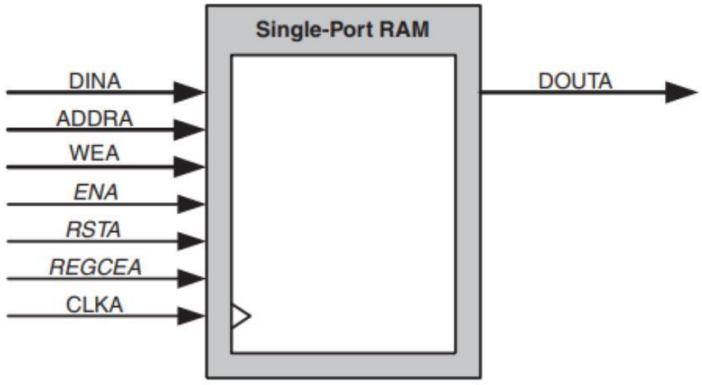

Fig: 2.5 Single Port RAM Memory type

COE files are used to initialize memory in ROMs. In Other Options, submit the memory initiation (COE) file. The .veo files are created in the project's folder during the ROM generation process. The contents of the.veo files provide the port number names needed to initialize the ROM memory.

2.4 VGA Interface and Image Buffer

The Video Graphics Array (VGA) standard is a simple method for showing graphics on CRT as well as most LCD computer displays. The protocol entails "scanning" the display and using two signals, hsync and vsync, to synchronize the precise place on the screen when the cursor is prepared to draw. The user guide describes the technical aspects of signal generation for the VGA standard. We will need to study these Xilinx Spartan-3E user manual pages to better understand what you will be working with in this lab. VGA timing data can be obtained at VGA. Timing and an VGA resolution calculator are available at VGA To RGB.

The FPGA board includes a typical VGA connector and communicates with the display via five signals. The signals are: hsync, vsync, blue, red, and green. With these three signals that are we are able to produce eight different colors, so it's certainly limited but useful for educational purposes. A VGA Drivers is typically little more than logic that turns on and off both vertical and horizontal sync signals so that the scan gun beneath the display knows

when to shoot the next row per column**.**

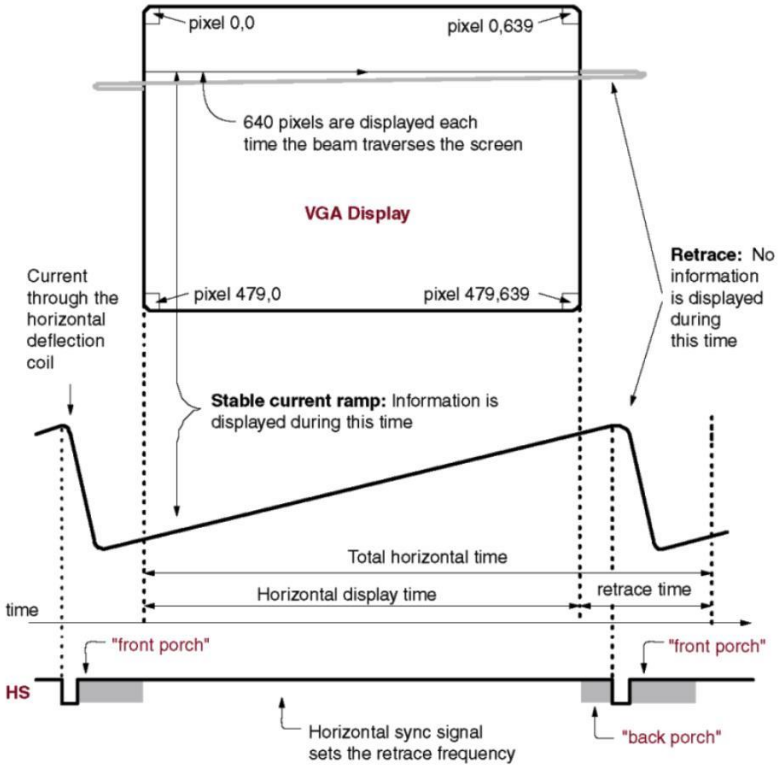

Fig.2.6 Image showing the Hsync

2.5 Image Data Processing and Edge Detection

Edge detection is a basic image processing technique that identifies and locates the borders or edges of every object in an image. It is used to recognize and identify discontinuities in image intensity, as well as extract the outline of objects in an image. The edges of any item in an image (for example, a flower) are commonly characterized as regions with an abrupt change in intensity. The purpose of edge detection was to highlight these places.

Edge detection techniques include the following:

- Sobel Edge Detection[16]

- Canny Edge Detection[15]

2.5.1. Sobel Edge Detection

Sobel edge detection represents a prominent image processing along with computer vision approach for recognizing edges in images. It is a gradient-based approach that calculates the gradient magnitude as well as direction for each pixel in the image by performing convolution operations with particular kernels.

The first stage is to capture picture data, which is typically in RGB format as well as must be converted into grayscale enabling edge detection. A grayscale filter calculates a single value for intensity by averaging each pixel's blue, red, and green components. This can be accomplished by extracting including averaging RGB values with bitwise operations including shifts. For example, extracting the blue, red, and green parts requires right-shifting as well as masking operations. After averages, the values combine to form one 8-bit grayscale result.

The Sobel operator calculates the gradient of image intensity for each pixel using two 3×3 kernels. These kernels emphasize edges in both horizontal and vertical orientations. Convolution on the image using these kernels yields two gradient parts: Gx (horizontal) as well as Gy (vertical). The amount of the gradient is subsequently computed, typically using the equation |Gx| + |Gy| to prevent the complexities of the square root calculations. This magnitude indicates the strength of an edge at each of the pixels[16].

To detect whether a pixel constitutes a component of a the edge, a thresholds operation is performed on the gradient magnitude. If the value of the magnitude exceed a predetermined threshold, a pixel is considered an edge; alternatively, it is not. This binary product is subsequently output, usually to a display as well memory. The Verilog code underlying this procedure consists of modules that execute convolution, determine gradient magnitudes, implement the threshold, then output the results. A Sobel module, for example, could accept pixel values, compute gradients, and then output binary edge mapping.

2.6 State of Art

The current state of the art in FPGA-based image processing in real time is defined as:

- **Deep Learning Speed**: Efficient execution of CNNs with object detection models such as YOLO on FPGA, resulting in low latency and high throughput.

- **Neuromorphic and Events-Based Processing**: The use of event cameras with neuromorphic algorithms enabling asynchronous data processing, resulting in energy-efficient real-time solutions.

- **Advanced FPGA Platforms**: The deployment of FPGA-based visual platforms capable of handling high-resolution image data streams in real time while being scalable and energy-efficient.

These developments make FPGA-based systems an appealing choice for actual time image processing jobs that demand great performance with minimal latency.

3

CIRCUITLEVELDESIGN

In the previous chapter, we have discussed the literature review and the details about the components in the circuit and in the upcoming chapter, we will see the implementation of the design with verilog . In this particular chapter, initially, we will discuss the block diagram of the design and the function of the sub-blocks in the given block diagram.

The remaining chapter discusses about the design of each block, the different types of filters available and the architecture chosen for the implementation. Followed by circuit level schematic simulation with the help of Xilinx Vivado are also discussed.

3.1 Top Level Circuit

We have chosen Basys3 for the implementation. Fig. 3.1 depicts the block diagram of the processing of image data for the tool box.

The basic blocks of the tool box are

1. Memory Instantiation(.coe file storage)
2. Image Processing Unit
3. VGA Timing Generator.

In this thesis, our primary focus is on the implementation of FPGA based image data processing for the implementation of the various filters.

The technology used for the implementation was AXI protocols technology. Xilinx Vivado is a system-based solution that provides ability to do simulations, and provide proper processing of the image data and its schematic layout.

After getting the output, we are able to see the processed image data which is obtained after filtering. To implement different image filtering and detect edge algorithms on a Field Programmable Gate Array (FPGA) using Verilog, alongside image data stored in a.coe file loaded onto block memory (BRAM), a top-level circuit layout typically includes several key elements: memory interactions, image processing components, control logic, as well as output interfaces. The following is an explanation of how all of these elements are organized and interact.

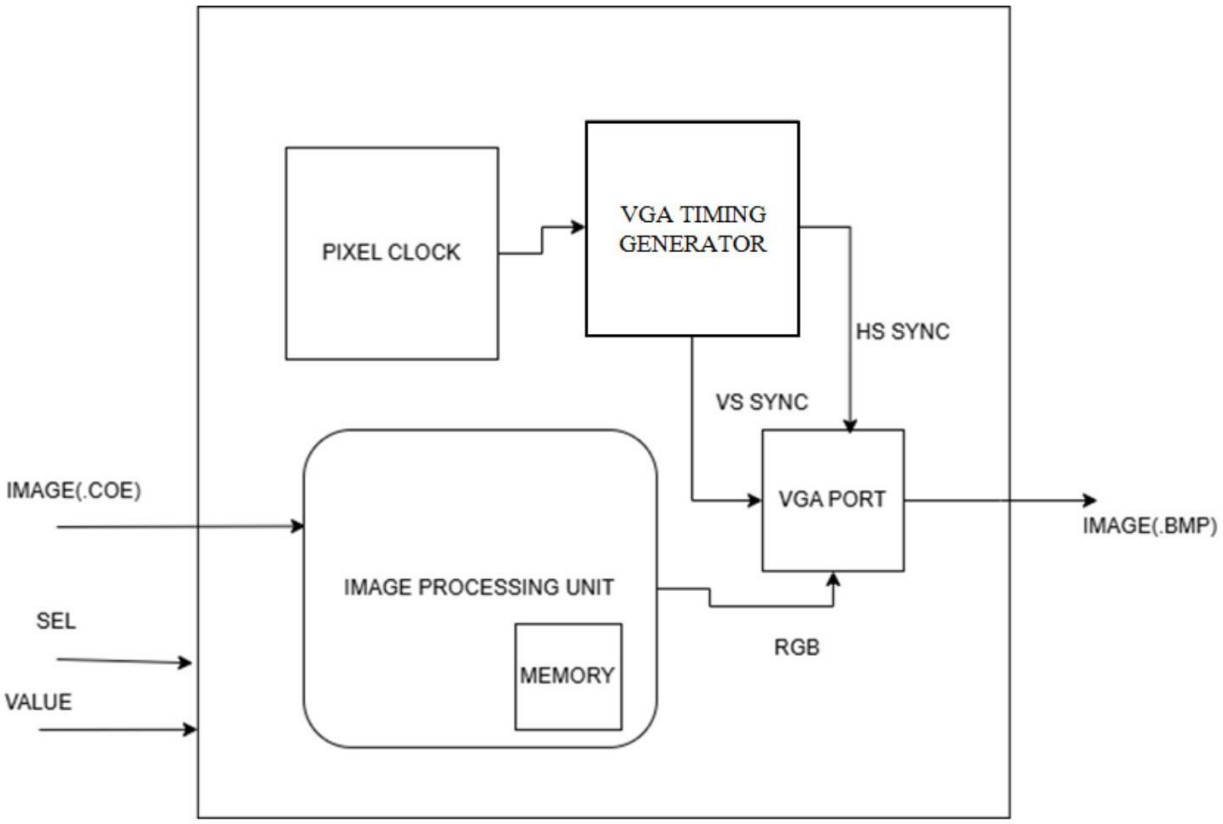

Fig.3.1.Block Diagram of the FPGA Implementation

The above block diagram indicates the data flow for the image information. This Diagram indicates the processing of bytes and how the input select line acts as the selection of the filters where various data filtering techniques are implemented.

As we can see the implemented diagram has a pixel clock which is used for the clocking of the circuit which is important for the timing of the data and also the VGA timing circuit which is used for counting the hysnc and vsync with the counter inbuilt to count the number of the pixels and set the end of the frame as well as end of the line of the image. Also the image processing unit also has the data storage which include the memory unit like BRAM which is responsible for providing the RGB data inside.

3.1.1 Explanation of the Block Diagram

This block diagram depicts a VGA-based processing system that accepts an input image (.coe format), analyses it, and displays it on a VGA display as a.bmp image. Below is a thorough description of each block

Overall flow:

- **Input:** IMAGE(.COE) - an image file in COE format (for initializing memory in FPGA systems).
- **Output:** IMAGE(.BMP) - shown using a VGA port.
- **Control Inputs**: SEL as well as VALUE are used to configure and control image processing operations

Blocks Description:

i) **Image (.COE) :**This input provides image data in.coe format, which is frequently used to populate memory in FPGAs.It's fed into the Imaging Processing Unit.

ii) **Image Processing Unit** : The core component responsible for picture storage, modification, and processing.

- MEMORY - holds the pixel data from the input image.
- It does actions such as augmentation, filtering, or any other algorithm based on SEL and VALUE.
- SEL selects a process (for example, grayscale or edge detection).
- VALUE could represent an operation parameter (for example, a threshold).

Executes algorithms for processing the stored image, such as:

- Grayscale conversion

- Edge detection (e.g., Sobel, Canny)
- Thresholding
- Brightness/Contrast adjustment
- Color inversion

iii) **Pixel Clock**: creates a clock signal to synchronize pixels, used to align image data creation with VGA timing.

iv) **Timing Generator for VGA:** produces the necessary vertical ,synchronization (VS SYNC) and horizontal synchronization (HS SYNC) signals.VGA displays require these timing signals in order to determine when to begin a new line as well as frame.

v) **Port VGA**: Transforms processed picture information into the signals that a VGA display needs, with

Inputs: RGB information obtained from the image processor. VGA timing generator signals for VS and HS synchronization.

VGA-compatible signal output: a picture on the screen that looks like a.BMP file.

3.2 Image File Generation(.Coe file)

The first block important for the processing as to convert the image which is in form of jpg to .coe file . The data which can be instantiated in RAM which is BRAM or distributed RAM in the FPGA. So, the following section discusses the design chosen for implementation.

3.2.1 Opencv Python Scripting

The Opencv python Scripting is used to check whether the image is in jpg

format and then retrieve the pixel data.

1. **coe_generator.py**:

The goal is to create a.coe file from a regular image (such as a BMP, PNG, or JPG).

Use Case: For simple image loading, where neighboring pixel information and convolution are not required. The result is a.coe file containing sequentially ordered pixel values.

Example: Used to simply display the image in VGA without any additional processing instructions.

2. kernel_coe_generator.py:

The goal is to convert grayscale photos and add neighboring pixel data so that they can be used for convolution procedures such as edge detection blurring, making a sharp edge . In order to store the center pixel and its neighboring (3x3 or 5x5) pixels for use

In filters, it reads the values of the pixels.

Application: sophisticated image processing techniques such as object recognition or filtering.

3. Parallel_image_generator.py:

The results from coe_generator.py and kernel_coe_generator.py are combined into a single final.coe file for this purpose, enables FPGA parallel processing, when convolutional and pixel data are provided at the same time.

Output: A.coe file that is prepared for loading into the BRAM and other memory of the FPGA.

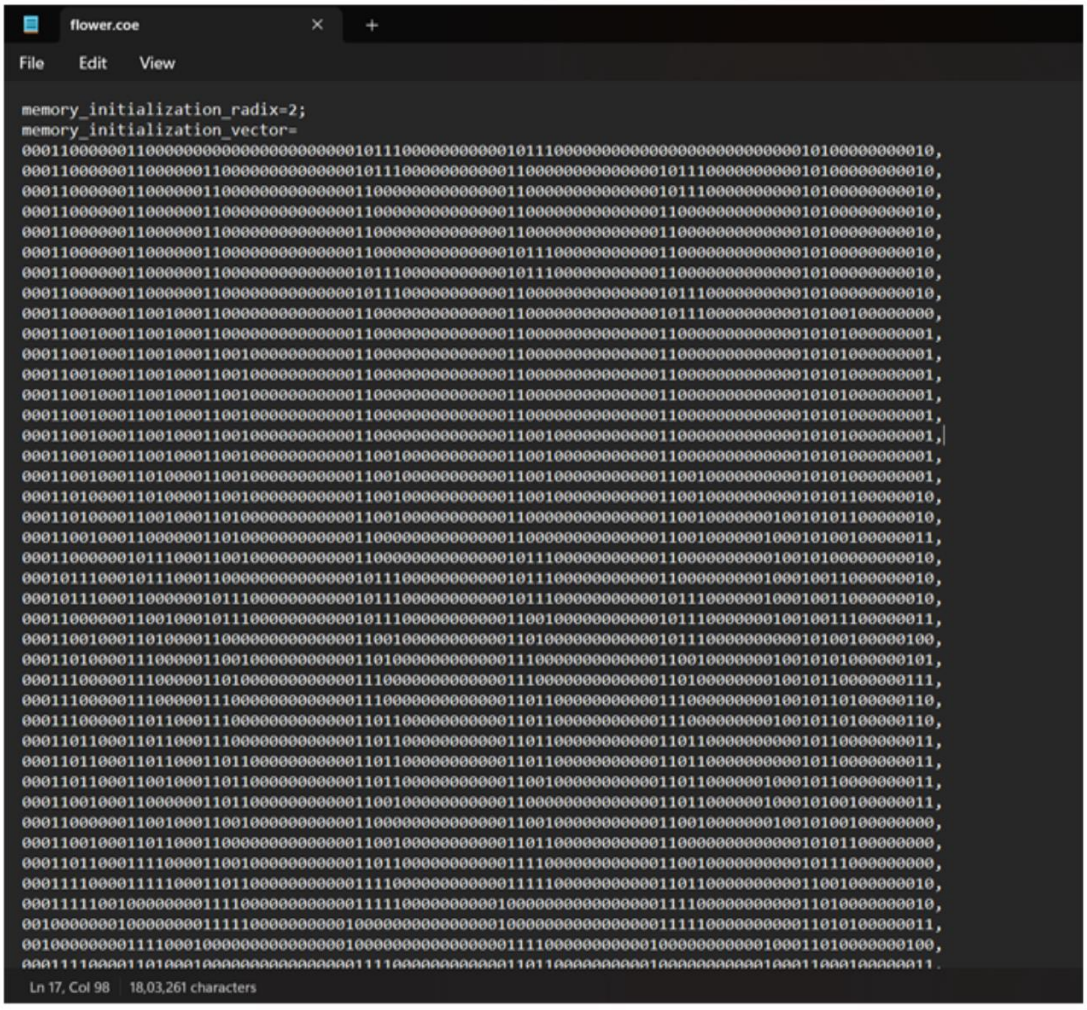

Fig.3.2 Generated .Coe File of the Input Image

3.3 Creating RAM/ROM in Vivado (Memory instantiation)

Here are some steps for utilizing Vivado to create block RAM or ROM. The screenshots were taken with Vivado 19.1 on Windows 10. Single or twin port memory creation is an option. Only one port of memory is needed in the majority of applications. Using IP Catalog, add a new source to your project and choose "Block Memory Generate" [Click]. The silicon inside an FPGA that is specifically designed and optimized for memory creation is called block memory. When performance is required, distributed memory uses flip flops

to construct memory, but this takes up a lot of space and resources on the chip[18].

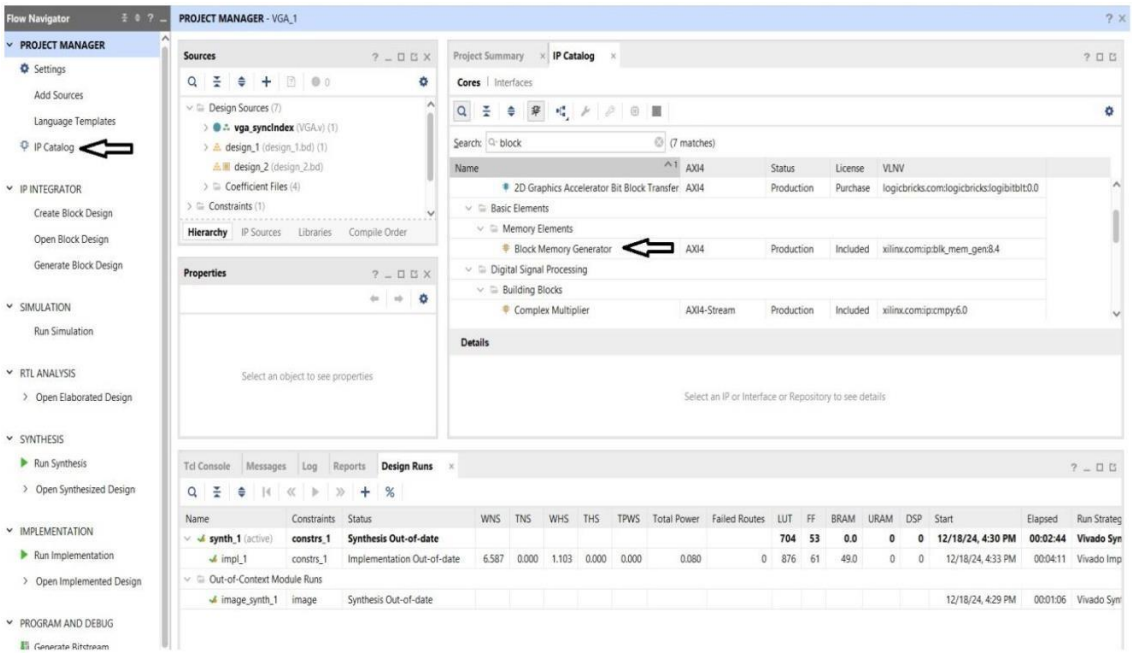

Fig .3.3 Configure IP Catalog in Vivado

In your project folder, create one single port ROM and give it the filename .

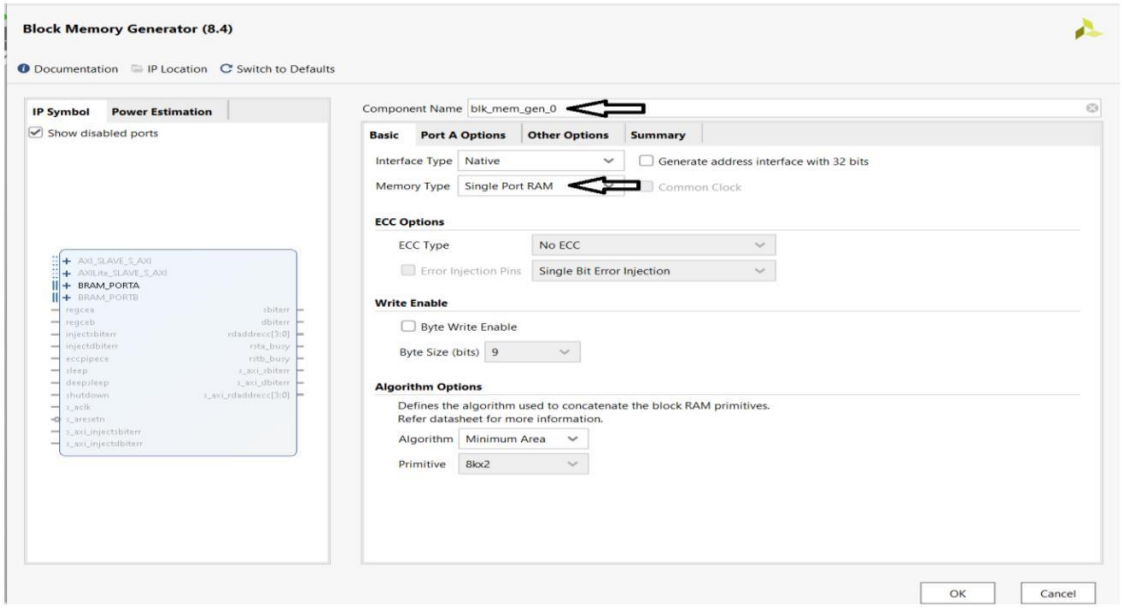

Fig .3.4 Instantiate Memory in RAM

COE files are used to initialize the memory in ROMs. Provide the memory initization (COE) file and select "Load Init File" under Other Options.

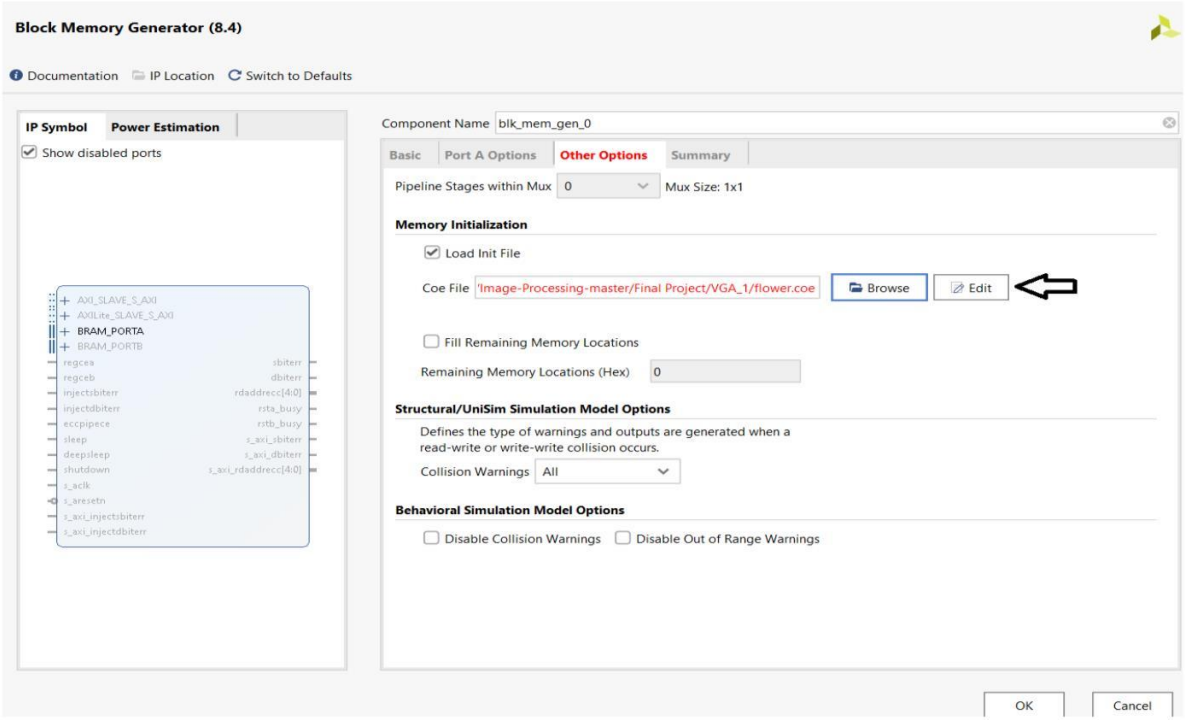

Fig.3.5.Load the generated .Coe file

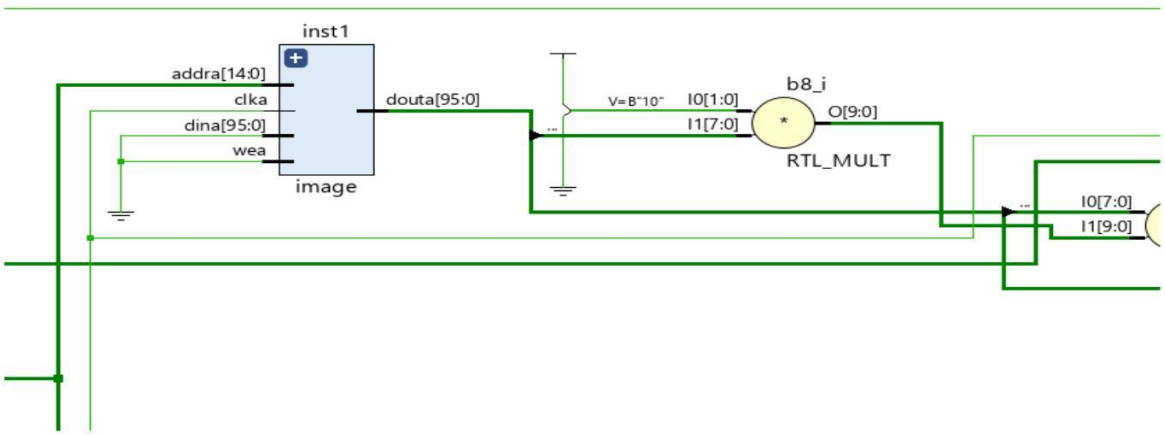

Fig.3.6 Image module Instantiation

3.4 Image Processing Unit

The VGA-based imaging processing system's synthesized hardware architecture is represented by the provided circuit. The main inputs for this architecture are a clock signal allowing synchronous operation, a select line (SEL) for selecting the image processing procedure, and the image data in.coe format, that is loaded into memory. The system controls the data associated with the image and produces the relevant RGB output signals based on the processing function that has been selected. The design also generates vertical sync (VSYNC) and horizontal sync (HSYNC) signals, which are necessary for the display of a VGA monitor to be properly synchronized.

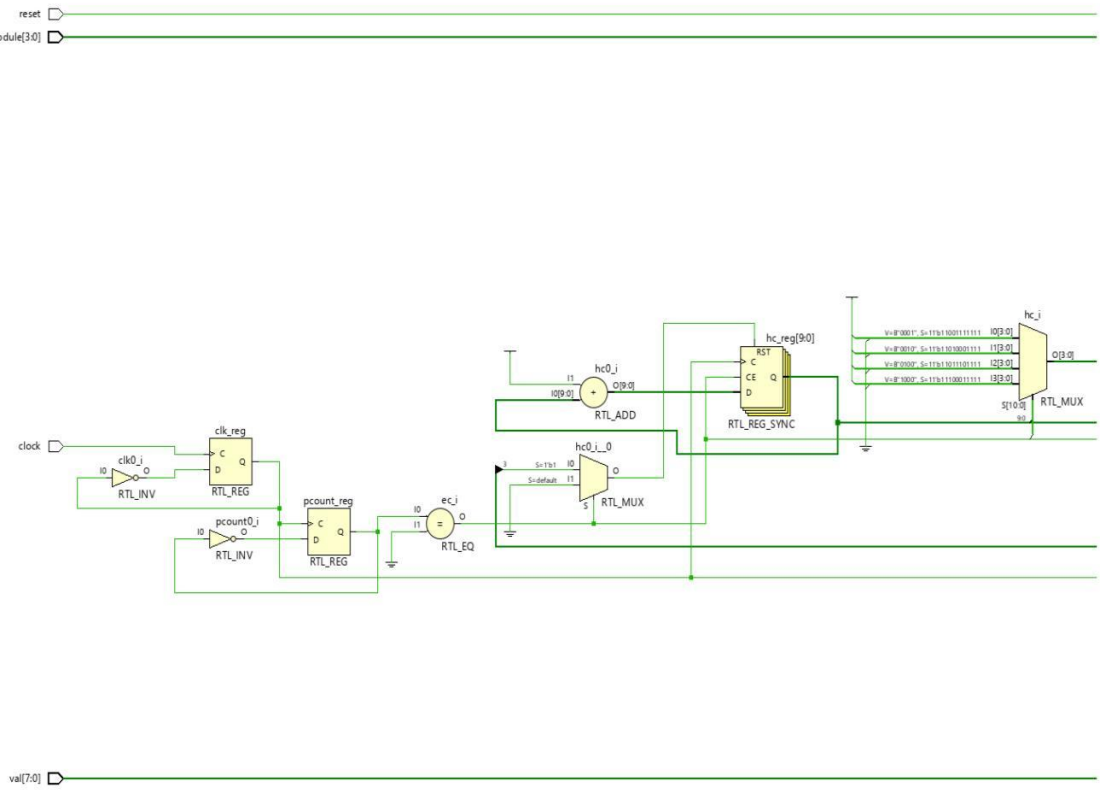

Fig.3.7. RTL Schematic of inputs

The Above part of the schematic pictures the input sent in the form of image data from the coe file and the Fig3.6 indicates the zoomed version of the inputs in RTL schematic where indicates the select line, value, clock ,reset lines. The Below Fig 3.7 indicates the processed output which is RGB data Lines and the vsync and hsync.

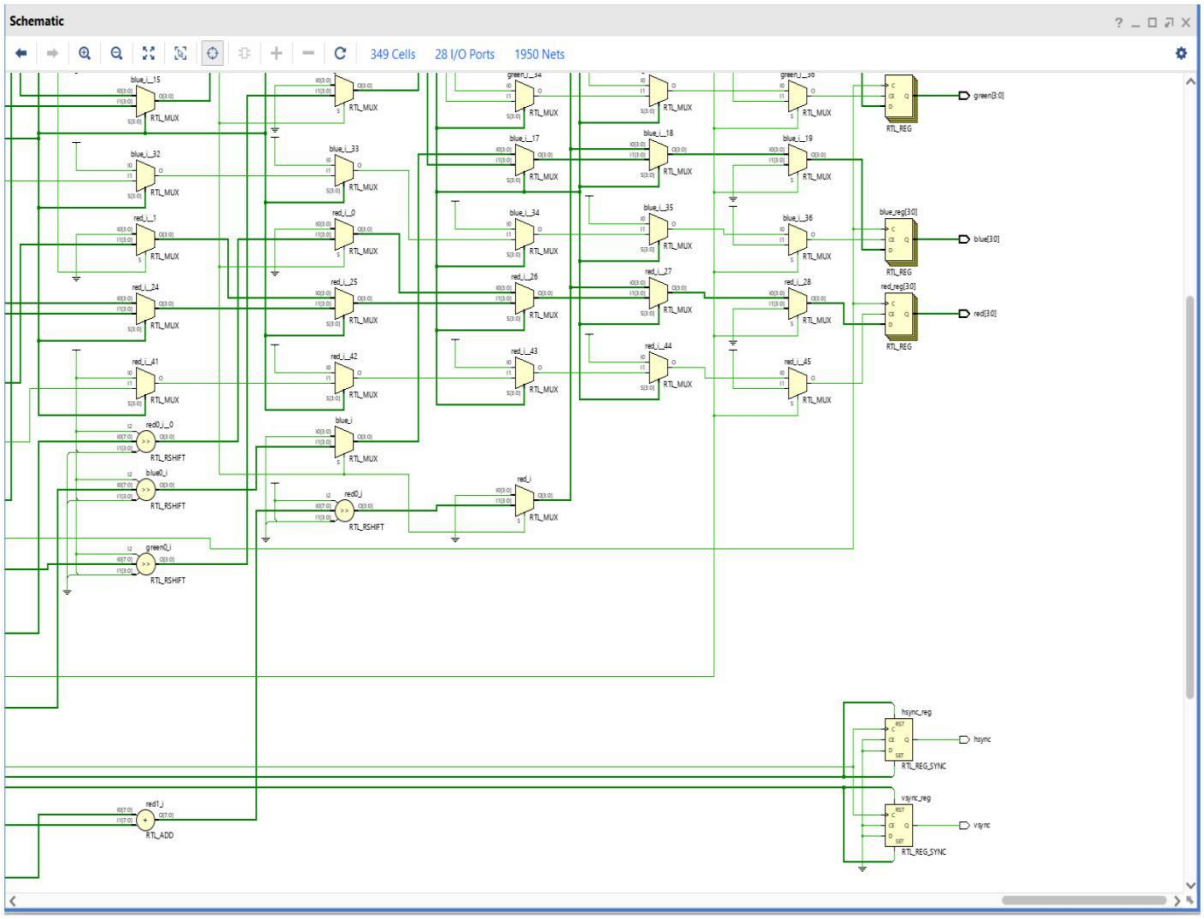

Fig 3.8 RTL Schematic of the outputs showing R,G,B Channels

As the input select line is selected which is of 4 b'0001(4 bit) the required filtering and application of the edge detection is performed and the data is sent out in the form of R,G,B channels. Below is the full RTL schematic representation of the tool box including the inputs and the outputs.

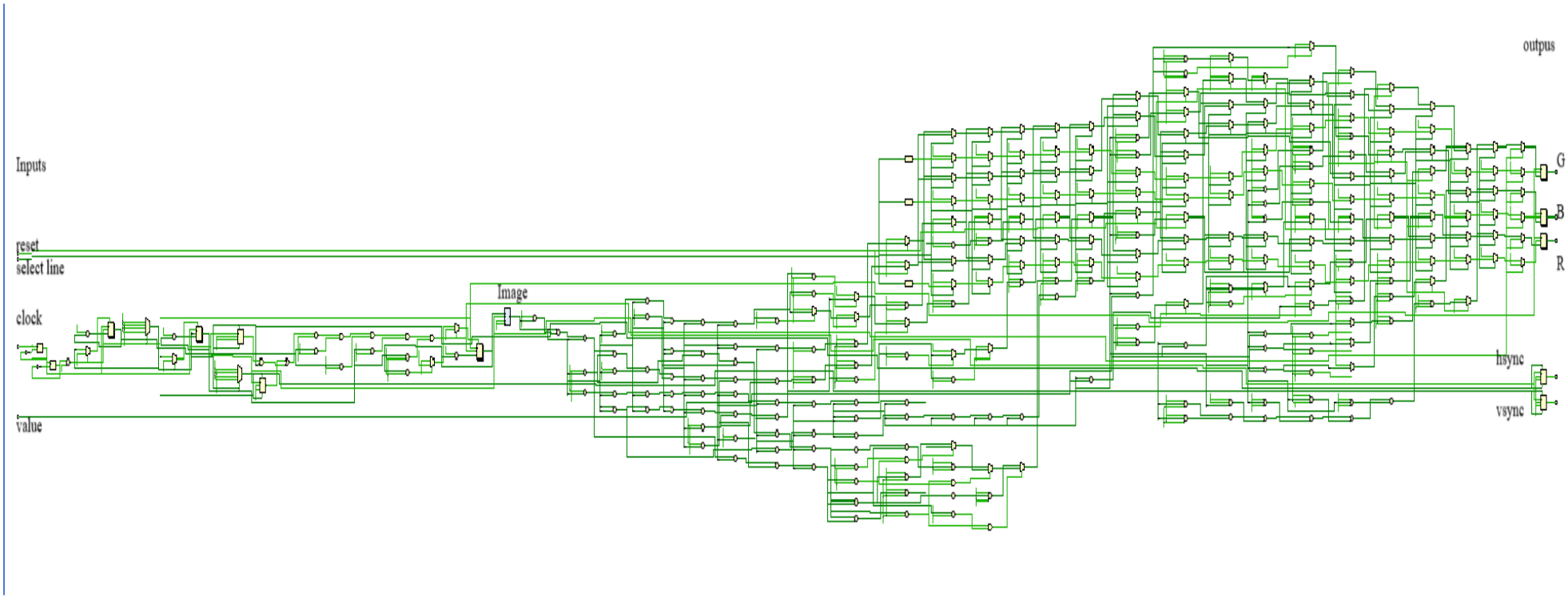

Fig.3.9 Complete RTL Schematic of the design

4

IMPLEMENTATIONANDRESULTS

Previously we have discussed the implementation of the block diagram for the pipeline of the image data.

This chapter will now show the synthesized and implemented design along with the power analysis and bit stream file generation, following that, the outcomes are examined.

4.1 Circuit Level Design Results

Fig 4.1 shows the synthesize of the design implemented in Vivado Xilinx.

Table4.1depicts the input parameters of the image and specifications of the Board.

Table4.1 Input Parameters for Image

Parameter	Value
Technology	Opencv python,Xilinx Vivado
image	JPG
Image size	480x640p
Refresh rate	60hz

The below table indicates the input on the select line with which the filters like increase in brightness, decrease in brightness, gray scale converting and some edge detections

4.1.1 Synthesis Design Implementation

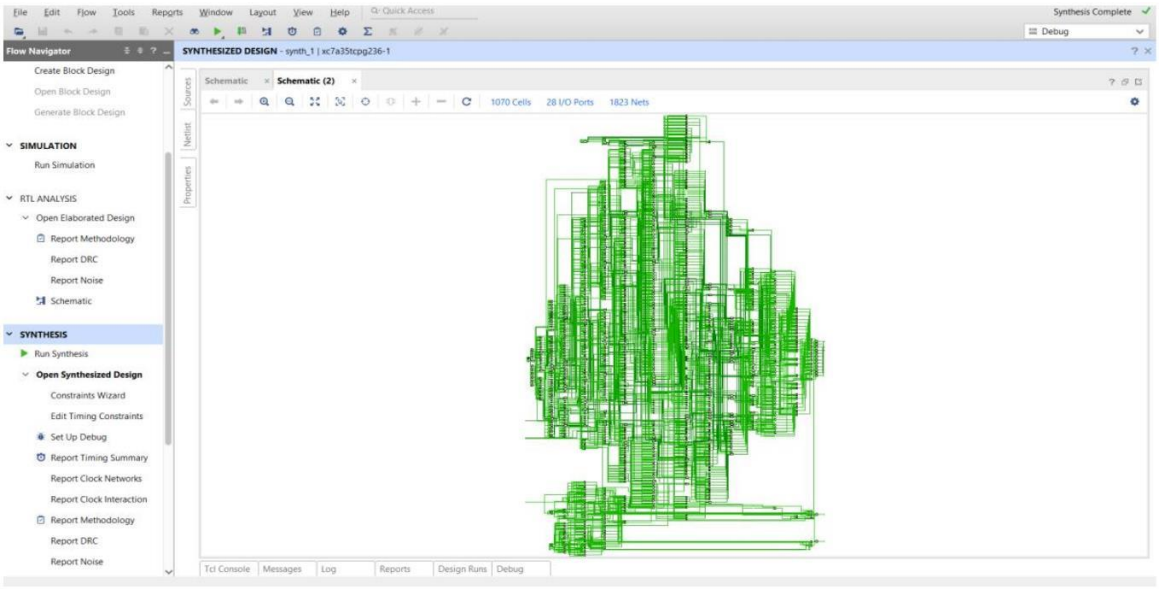

Fig4.1 Synthesized design of the RTL

4.1.2 Bit file Generation

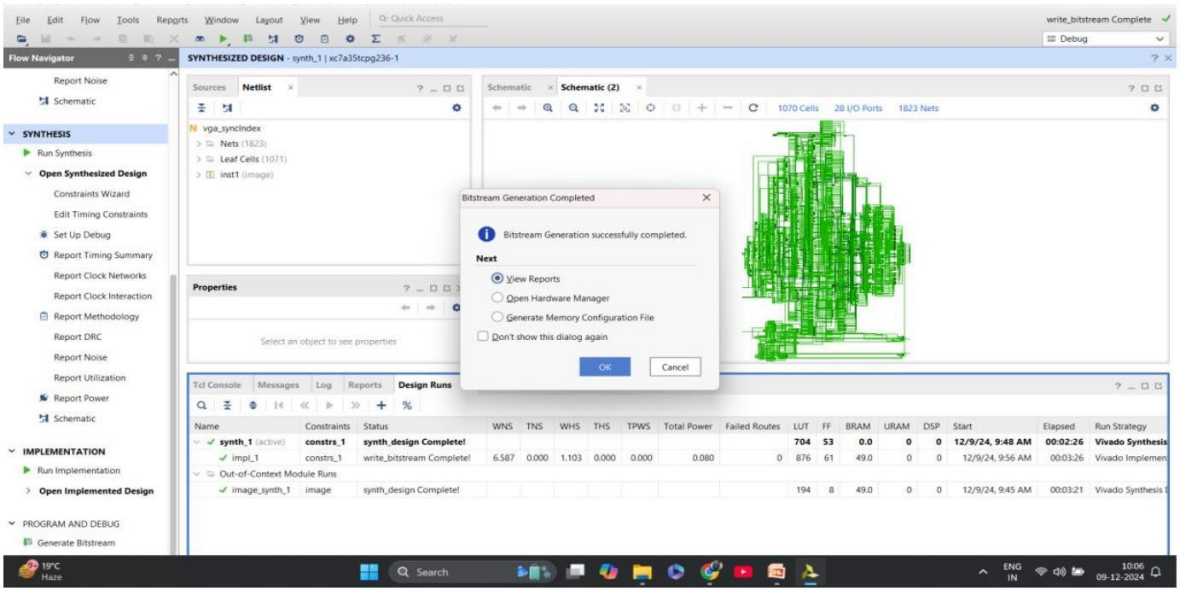

Fig4.2Sucessfull generation of the bit file

Fig4.2 shows the bit file is generated for the hardware connecting and successful implementation of Basys 3 board.

4.2 Output simulations Based upon the Select line

Table4.2 depicts the inputs on the select lines and resulting filters.

Select line input	Filter
0000	RGB to Gray
0001	Increase brightness
0010	Decrease brightness
0011	Color Inversion
0100	Red Filter
0101	Blue Filter
0110	Green Filter
0111	Original Image
1000	Average Blurring
1001	Sobel Edge Detection
1010	Edge Detection
1011	Motion Blurring
1100	Emboss
1101	Sharpen
1110	Motion Blur
1111	Gaussian Blur

Table 4.2 Input selects and Corresponding Filters

4.3 Simulation Results on the Basys 3 Board

Below are the results indicating the execution of the filtering of the image based upon the 4 bit select line which is selected on the pins of the Basys 3 board showing the processed image on the Desktop Display which is shown in the Fig.4.3.

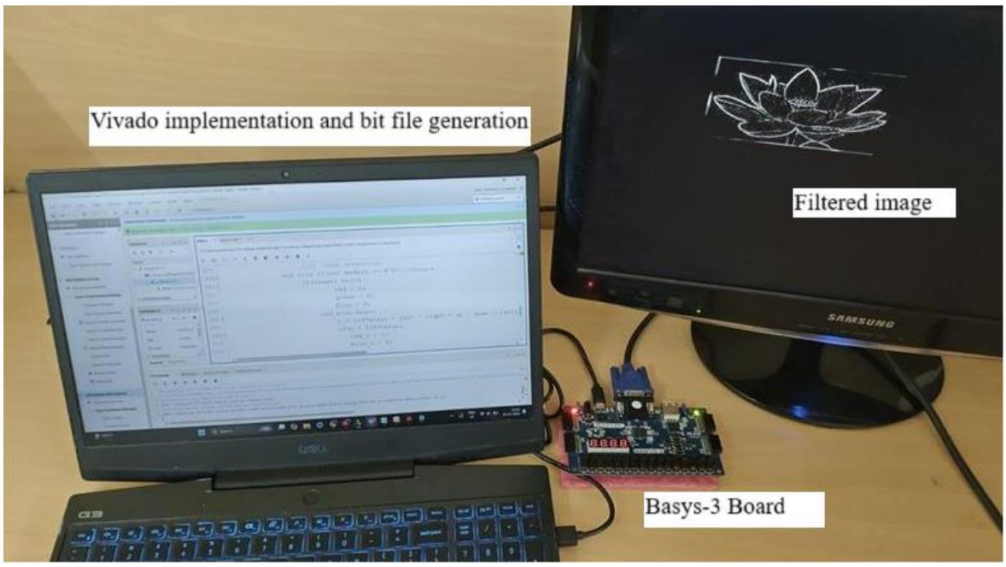

Fig.4.3 Basys-3 FPGA Board Implementation and Bit file Generation

4.4 Results :

i) Rgb to gray scale(0000):

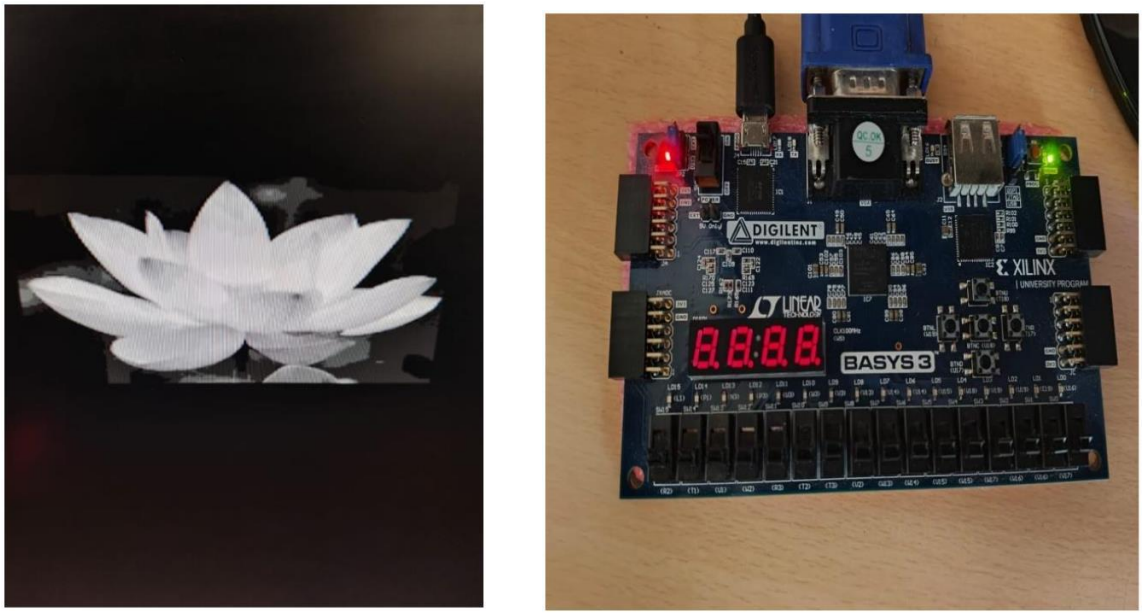

Fig 4.4 Rgb to Gray Scale Converstion

ii) Increase brightness(0001):

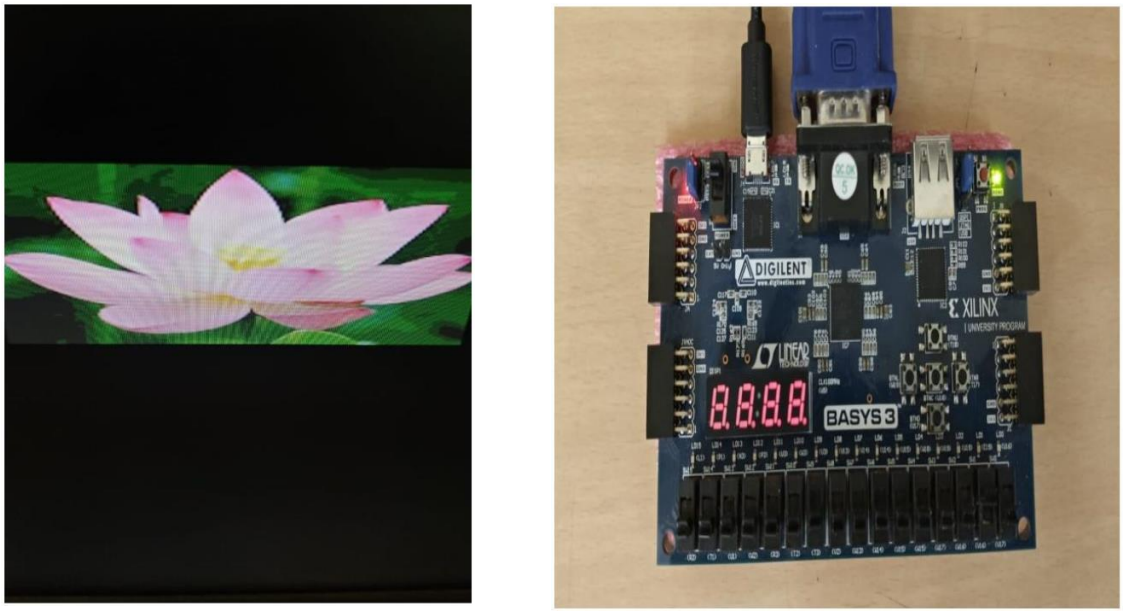

Fig.4.5 Increase in the Brightness of Image

iii) Colour Inversion(0011):

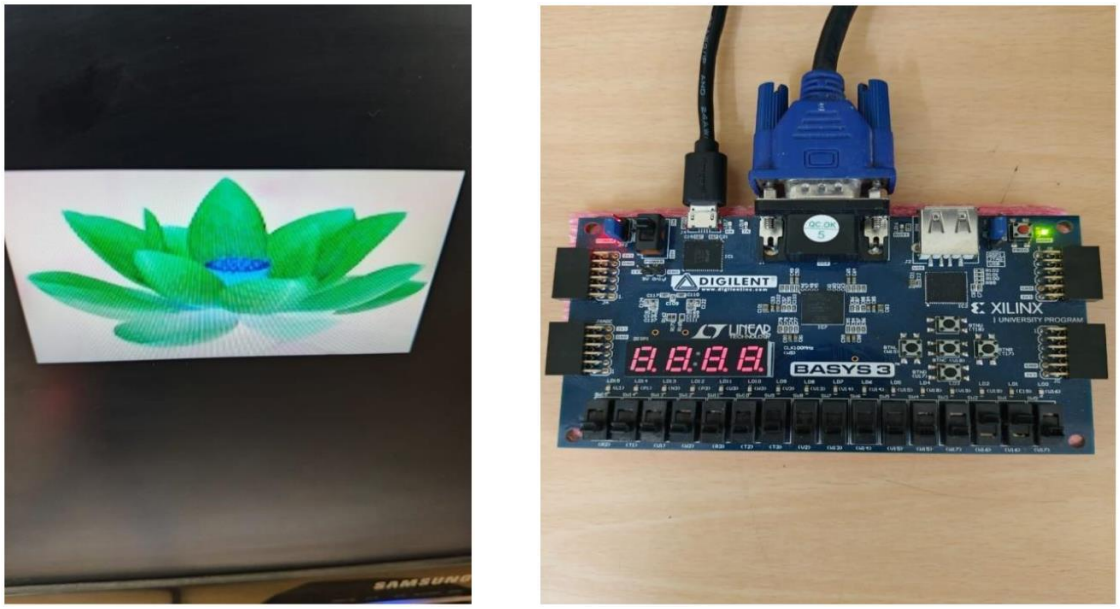

Fig 4.6 Colour inversion Filter

iv) Average Blurring(1000):

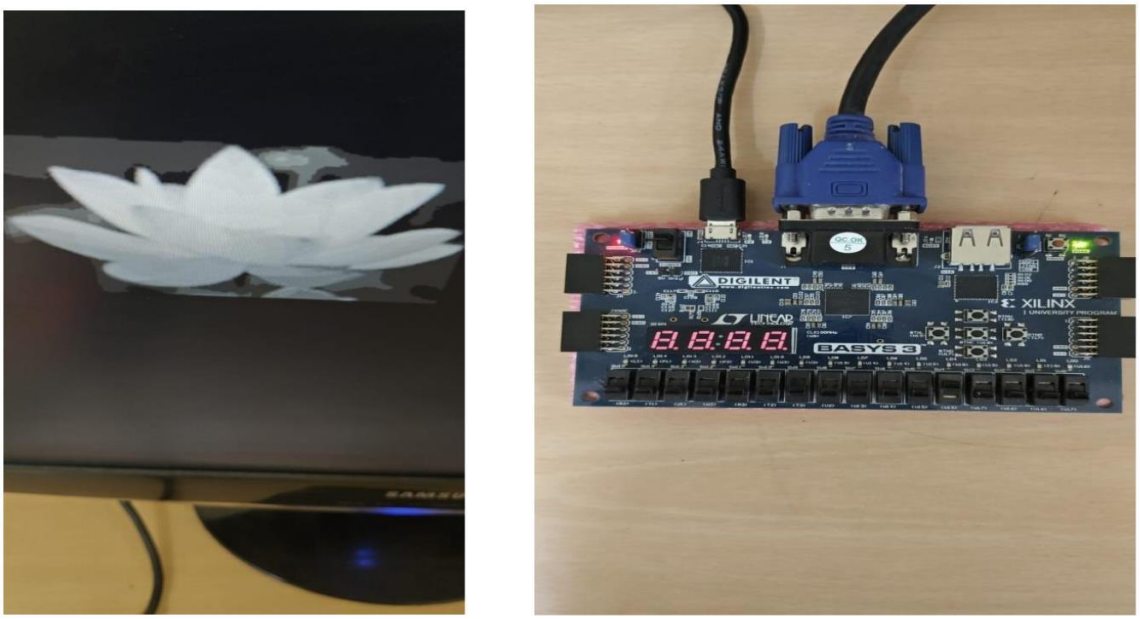

Fig 4.7 Average Blurring Filter

v) Emboss (1100):

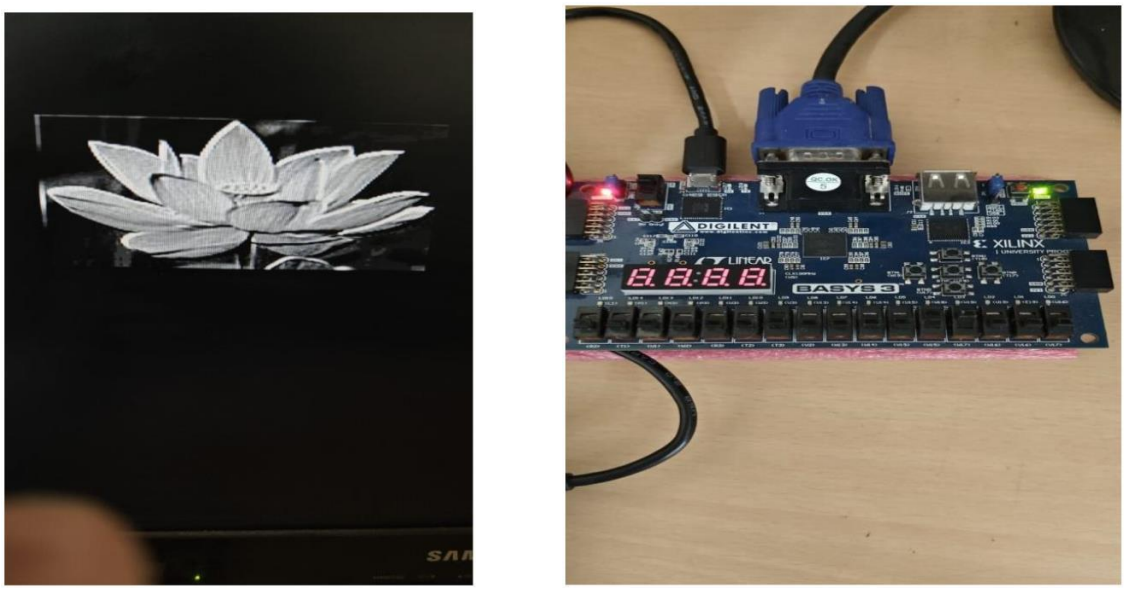

Fig 4.8 Emboss Filter

vi) Sharpen (1101):

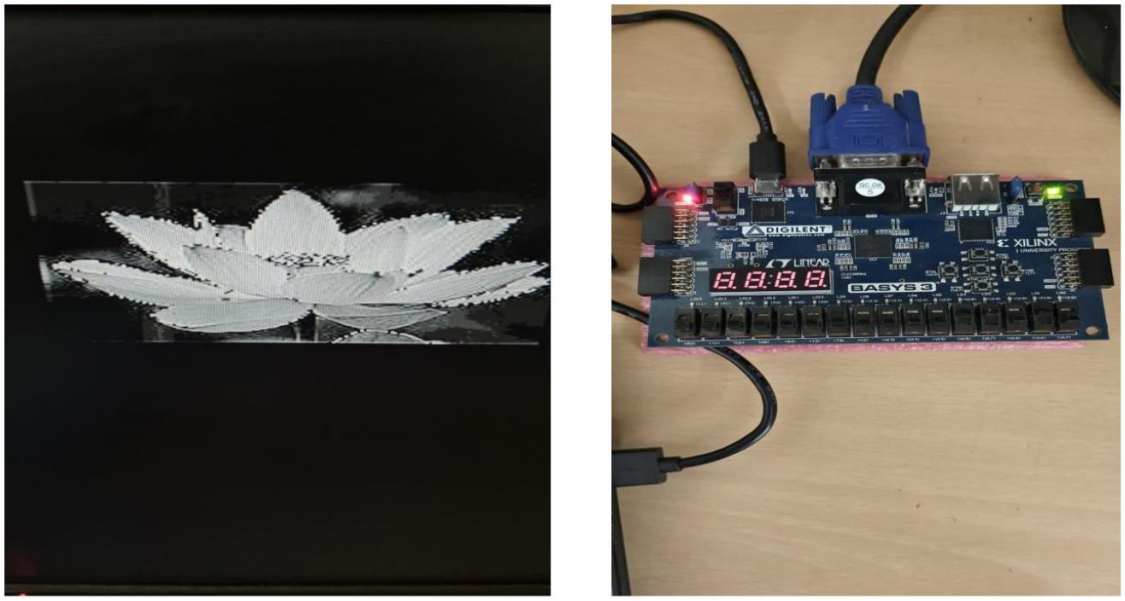

Fig 4.9 Sharpen Filter

vii) Sobel Edge Detection(1001):

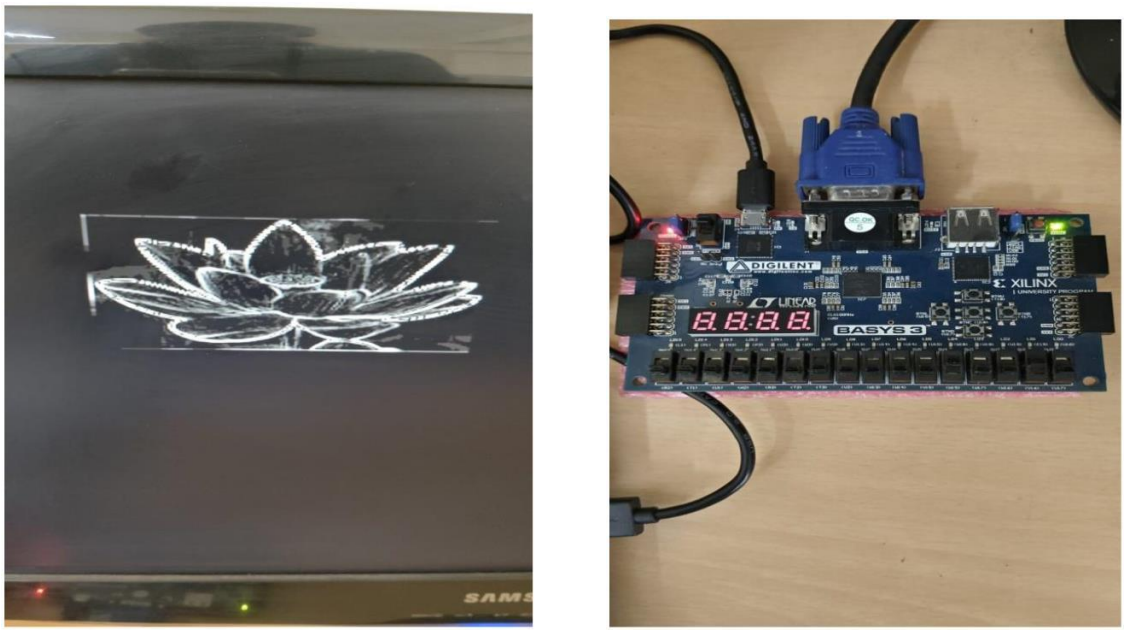

Fig 4.10 Sobel Edge Detection

viii) Motion Blur(1110):

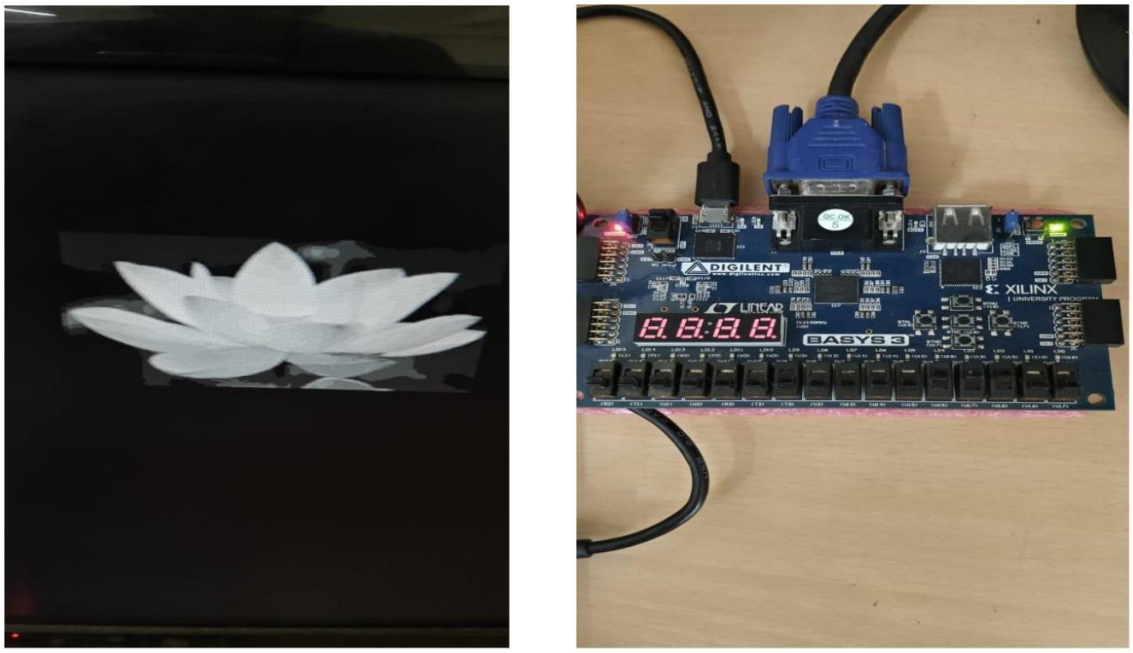

Fig 4.11 Motion Blur Filter

xi) Edge Detection(1010):

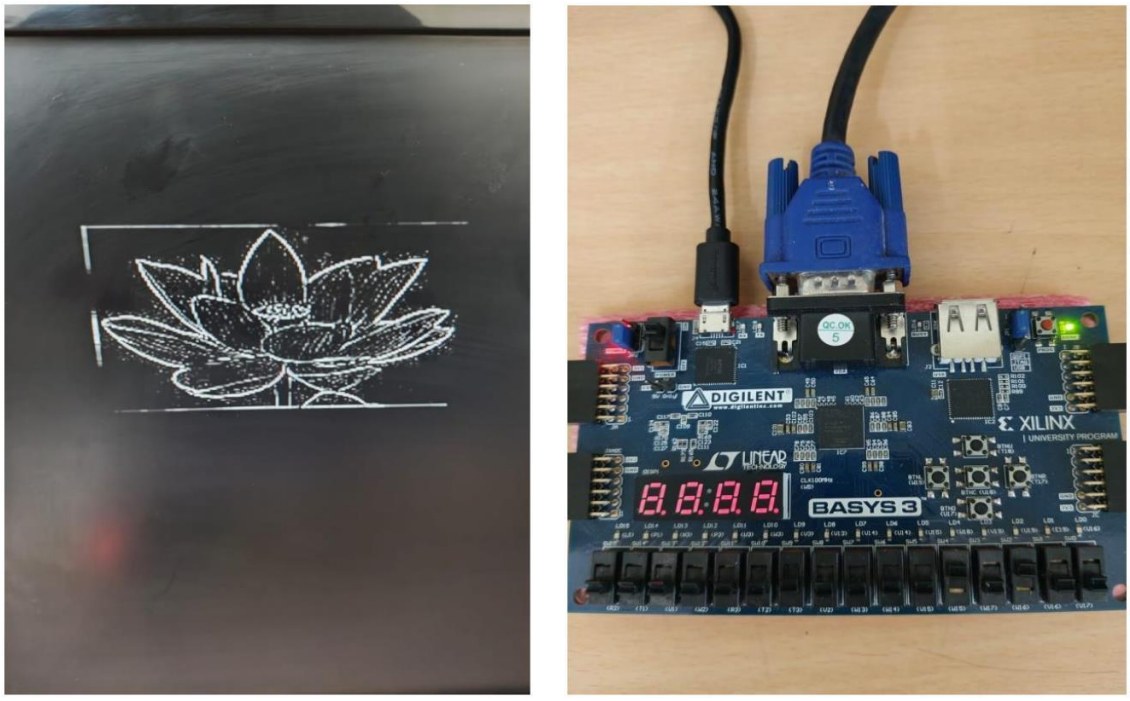

Fig 4.12 Edge Detection Filter

4.5 Power Analysis and Summary

Power analysis of the whole design is done on the circuit. After analysis, it was seen that the power consumed by the full circuit with total on chip power is 0.08W

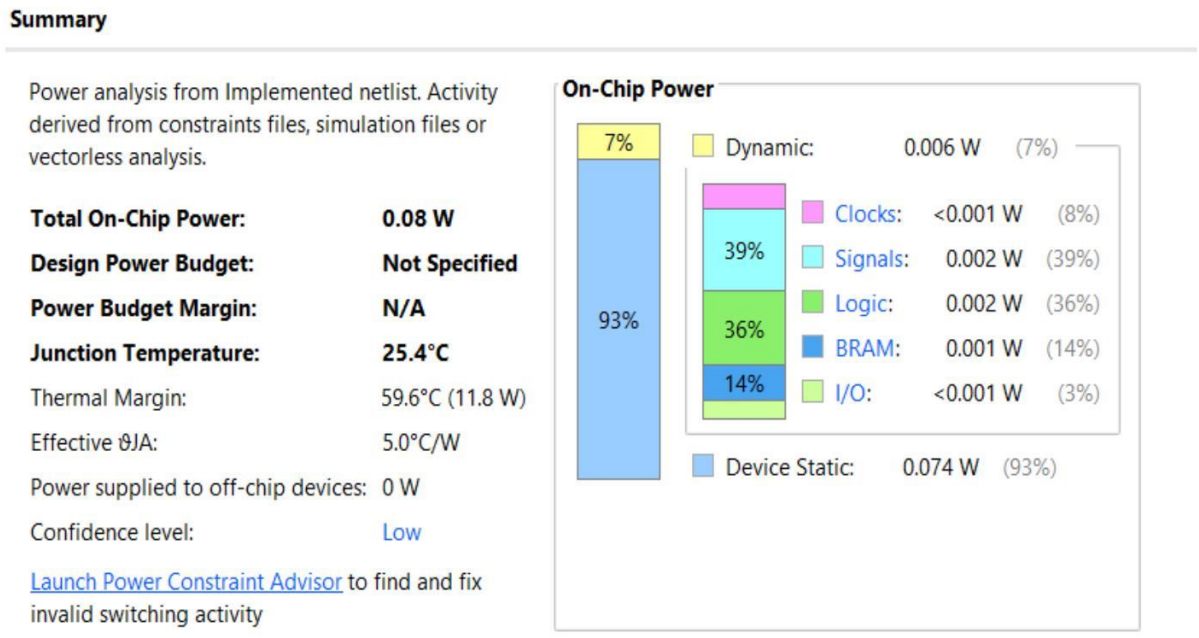
Summary
Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.
Total On-Chip Power: 0.08 W
Design Power Budget: Not Specified
Power Budget Margin: N/A
Junction Temperature: 25.4°C
Thermal Margin: 59.6°C (11.8 W)
Effective ϑJA: 5.0°C/W
Power supplied to off-chip devices: 0 W
Confidence level: Low
Launch Power Constraint Advisor to find and fix invalid switching activity
On-Chip Power
7%
93%
Dynamic: 0.006 W (7%)
39%
36%
14%
Clocks: <0.001 W (8%)
Signals: 0.002 W (39%)
Logic: 0.002 W (36%)
BRAM: 0.001 W (14%)
I/O: <0.001 W (3%)
Device Static: 0.074 W (93%)

Fig4.13 Power contribution and report on Design

5

CONCLUSION AND FUTURESCOPE

5.1 Conclusion

This project shows a whole FPGA-based image processing pipeline, from PC image conversion to VGA output for real-time display. Images are transformed into.coe files that are compatible with Verilog's Block RAM setup using Python programs. Based on user inputs, the FPGA produces the image using a variety of methods, including color filtering, brightness modification, edge detection, blurring, sharpening, and grayscale conversion. Modular Verilog code is used to implement these processes, and HSYNC & VSYNC signals are used to synchronize outputs for correct VGA display. The Basys-3 FPGA platform implementation demonstrates the speed and efficacy of hardware-level parallelism in managing picture processing. Additionally, it highlights the educational benefits of using Verilog in digital system design and Python for data preparation. In addition to showcasing important image processing methods, this system lays the groundwork for real-time embedding vision systems in more extensive applications such as smart displays, robotics, and surveillance.

5.2 Future Scope

As the Advancements of the image processing with the refined data filtering with the real time applications they are considered for the various applications like the mechanical parts inspection with the cracks detection[2]

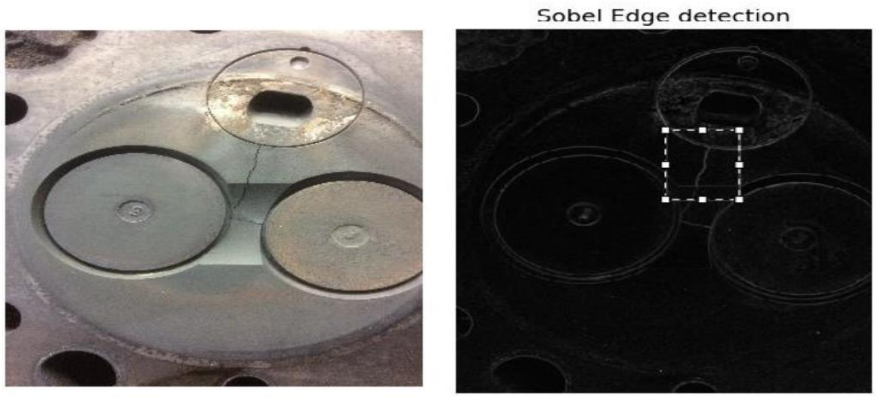

Fig 5.1 Edge Detection in mechanical parts inspection

And with the advancements in the automobile industry and the no human intervention vehicular systems there is a need in the lane detections which could be also an extension for the real time embedded systems processing[5].

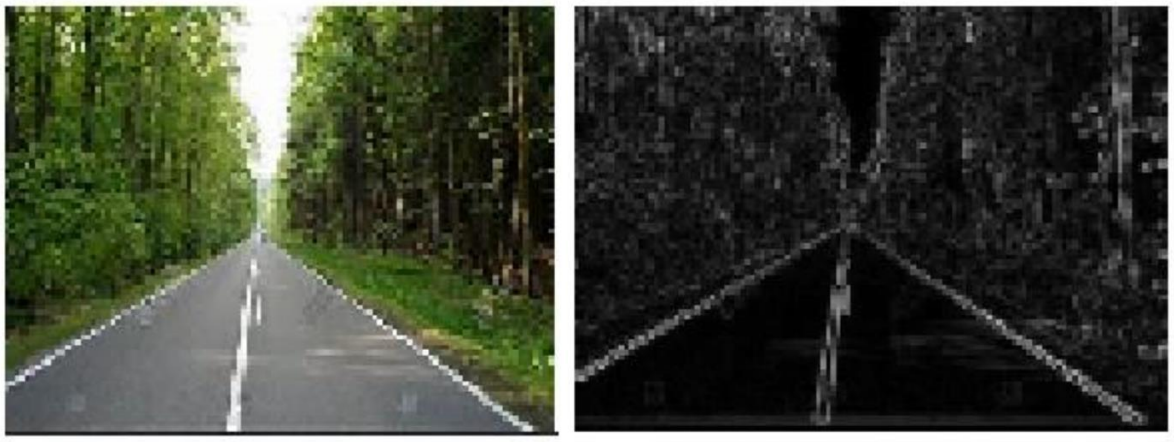

Fig 5.2.Lane Detection with the Edge Detection

With the encroachment of the autonomous surveillance has a huge future scope in the development of the real time automation.

6

REFERENCES

[1] Jen-Yi-Hsu,Paul C-P.Chao,Ting-Yin Jiang, 'A Fast FPGA Hardware Accelerator for Remote Heart Rate Detection Based on RGB Vision", in IEEE transactions on biomedical circuits and systems, vol. 18, no. 3, june 2024

[2] Lv Y (2024), Application of image processing technology based on field programmable gate array in mechanical part inspection. Front. Mech. Eng 10:1406559.doi: 10.3389/fmech.2024.1406559

[3] Dimitris Perdios, Florian Martinez,"CNN-Based Image Reconstruction Method for Ultrafast Ultrasound Imaging" in IEEE Transactions on ultrasonics, ferroelectrics, and frequency control, vol. 69, no. 4, april 2022.

[4] Ahmad Mouri Zdeh Khaki , Ankur Singh,"A Resource-Efficient Multi Function Embedded Eye Tracker System Implemented on FPGA" in IEEE Access as Nano-Material Technology Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and Information and Communication Technology,and in part by Hyundai Motor Company,Digital Object Identifier 10.1109/ACCESS.2023.3342839

[5] David Castells-Rufas,Marc Codina,Debore Gil,"A Survey of FPGABased Vision Systems for Autonomous Cars" in IEEE Access as work was supported by grants PID2021-126776OB-C21, RTI2018-095209- B-C21, RTI2018-095209-B-C22 funded by the MCIN from Spain with grant number MCIN/AEI/10.13039/501100011033" Digital Object Identifier 10.1109/ACCESS.2022.3230282

[6] Nandihini.M ,Faridha Banu D,Madhumitha,"Evaluating Edge Detection Algorithms for Robust Image Processing and FPGA Implementation",in 2025 5th International Conference on Trends in Material Science and Inventive Materials (ICTMIM) — 979-8-3315-0148-8/25/ IEEE — DOI: 10.1109/ICTMIM65579.2025.1098824

[7] Ming Xu,Lianng Chen , Hao Shi,"FPGA-Based Implementation of Ship Detection for Satellite On-Board Processing" in IEEE Journal of selected topics in applied earth observations and remote sensing, vol. 15, 2022

[8] Shuaiqing Zhi,yani Cui,Jiaxian Deng,"An FPGA-Based Simple RGBHSI Space Conversion Algorithm for Hardware Image Processing"in IEEE Access by Hainan Provincial Key Research and Development Plan, National Natural Science

Foundation of China and Macau Science and Technology Development,Digital Object Identifier 10.1109/ACCESS.2020.3026189

[9] Jin Woo Park,Boyeal Kim,Hyun Kim,"A Low-Cost and HighThroughput FPGA Implementation of the Retinex Algorithm for RealTime Video Enhancement",in IEEE Transactions on very large scale integration (vlsi) systems, vol. 28, no. 1, january 2020

[10]jayaraj U Kidav,Sreejeesh S.G,"An FPGA-Accelerated Parallel Digital Beamforming Core for Medical Ultrasound Sector Imaging", in IEEE Transactions on ultrasonics, ferroelectrics, and frequency control, vol. 69, no. 2, february 2022

[11] Jen-Yi Hsu,Ting-Yin Jiang,Paul C.,"A Fast FPGA Hardware Accelerator for Remote Heart Rate Detection Based on RGB Vision" in IEEE Transactions on biomedical circuits and systems, vol. 18, no. 3, june 2024

[12] Yuhai he,Jiye Huang,Yiming Pan,"A novel low-resource consumption and high-speed hardware implementation of HOG feature extraction on FPGA for human detection"in Science Direct,INTEGRATION, the VLSI journal 97 (2024) 102208,Received 2 December 2023; doi.org/10.1016/j.vlsi.2024.102208

[13] Gracieth Cavalcanti Batista,Johnny Oberg,Osamu Saotome,"Machine learning algorithm partially reconfigured on FPGA for an image edge detection system" in Journal of Electronic Science and Technology 22 (2024) 100248

[14] Harb, S., Ahmad, M.O., Swamy, M.N.S."An efficient image steganographic scheme for a real-time embedded system and its hardware implementation on AMD Xilinx Zynq-7000 APSoC platform". J RealTime Image Proc 20, 46 (2023).,doi.org/10.1007/s11554-023-01302-x.

[15] Mahalle, Avinash G., and A. M. Shah. "An Efficient Design for Canny Edge Detection Algorithm Using Xilinx System Generator." 2018 International Conference on Research in Intelligent and Computing in Engineering (RICE). IEEE, 2018.

[16] Nausheen, N., Seal, A., Khanna, P., Halder, S. (2018). A FPGA based implementation of Sobel edge detection. Microprocessors and Microsystems, 56, 84-91.

[17] Duvvuru, Abhiram Reddy, Kumar Sai Reddy Guntaka, and Shrikant Jadhav. "Design and Verification of an FPGA-Based MRI Image Cluster Processing."

SoutheastCon 2025. IEEE, 2025.

[18]Baali, Mehdi, et al. ”Implemention of Image Processing Platform on Low-cost ZedBoard FPGA Using Xilinx-Vivado.” 2024 International Conference on Advances in Electrical and Communication Technologies (ICAECOT). IEEE, 2024.

[19] Chen, Yutong, et al. ”Design of AXI Bus-Based IP Core for Image Processing.” CCF Conference on Big Data. Singapore: Springer Nature Singapore, 2024.

[20]Gtifa, Wafa, and Anis Sakly. ”Integrating Xilinx FPGA and intelligent techniques for improved precision in 3D brain tumor segmentation in medical imaging.” Journal of Real-Time Image Processing 20.6 (2023): 115.